

Data-driven Adaptive Stabilizer for Unknown Nonlinear Dynamic MIMO Systems Using a Cognition-based Framework

Von der Fakultät für Ingenieurwissenschaften,
Abteilung Maschinenbau und Verfahrenstechnik der
Universität Duisburg-Essen
zur Erlangung des akademischen Grades

einer

Doktorin der Ingenieurwissenschaften

Dr.-Ing.

genehmigte Dissertation

von

Xi Nowak

aus

Peking, V.R. China

Gutachter: Univ.-Prof. Dr.-Ing. Dirk Söffker
Univ.-Prof. Dr.-Ing. Steven Liu

Tag der mündlichen Prüfung: 04. August 2015

Danksagung

Die vorliegende Arbeit entstand während meiner Zeit als wissenschaftliche Mitarbeiterin und Doktorandin am Lehrstuhl Steuerung, Regelung und Systemdynamik der Universität Duisburg-Essen. In dieser Zeit habe ich nicht nur diese Dissertation abgeschlossen, sondern konnte neben dem Anfertigen von Forschungsarbeiten auch zusätzlich umfangreiche wissenschaftlichen und nicht wissenschaftlichen Kenntnisse erwerben, die für mein zukünftiges Leben sehr hilfreich sind. Daher möchte ich hier die Gelegenheit nutzen, all denjenigen zu danken, die mich dabei unterstützt haben.

Mein erster und besonderer Dank gilt meinem Betreuer Herrn Univ.-Prof. Dr.-Ing. Dirk Söffker für die Möglichkeit zur Promotion, für die unzähligen, vielseitigen, interessanten Gespräche und Diskussionen sowie für die erlebnisreichen gemeinsamen Dienstreisen weltweit.

Ebenfalls bedanke ich mich sehr bei Herrn Prof. Dr.-Ing. Steven Liu für die Begutachtung der Arbeit und die fachliche Diskussion in Kaiserslautern.

Weiterhin möchte ich mich bei meinen Kolleginnen und Kollegen am Lehrstuhl für die stets sehr erfolgreiche Zusammenarbeit, die tatkräftige Unterstützung, die vielseitigen freundlichen Gespräche und die schöne gemeinsame Zeit bedanken.

Mein größter Dank gilt meinen Eltern, ohne die ich den Weg nach Deutschland nicht gefunden hätte, und meinen Schwiegereltern, die mich stets motiviert und bestärkt sowie familiären Rückhalt gegeben haben.

Ganz besonders und von ganzem Herzen danke ich schließlich meinem Ehemann Michael Nowak, der mir immer helfend zur Seite stand, Motivator war und mir zu jeder Zeit Freiräume gegeben und Verständnis entgegengebracht hat. Außerdem möchte ich mich auch noch bei meinem kleinen Sohn Elias bedanken, den ich während der Anfertigung der Dissertation noch ganz ruhig in meinem Mutterleib spürte, der mir so die Zeit und Energie für die Arbeit gab und mich stets daran erinnerte, die Arbeit vor seiner Geburt rechtzeitig fertigzustellen. Ich freue mich sehr, dass ich nach der Promotion mehr gemeinsame Zeit mit ihnen verbringen kann.

Duisburg, im April 2016

Xi Nowak

Abstract

This thesis focuses on a cognitive stabilizer concept which is an adaptive discrete control method based on a cognition-based framework. The aim of the cognitive stabilizer is to autonomously stabilize a specific class of unknown nonlinear multi-input-multi-output (MIMO) systems. The cognitive stabilizer is able to gain useful local knowledge of the unknown system and can autonomously define suitable control inputs to stabilize the system.

The development of different kinds of adaptive, data-driven, and model-free controllers shows a clear tendency towards research on control methods with high autonomy. Here the term autonomy is used to describe the fact that the control approach/the related programming is organized such that the algorithm is able to handle the feedback design autonomously without instructions from outside the algorithm. Typical methods affected by this definition are adaptive control method, data-driven control method, and model-free control method. In this thesis, the state-of-the-art of them is reviewed. The main focus is given to the autonomy of the realized approaches. It can be concluded that the existing methods still show some open points achieving highly autonomous control. In order to address these open points, a framework similar to modeling approaches concerning the human cognition processes [Cac98] can be introduced in the engineering context, which is denoted as cognition-based framework. As stabilization control task is the most basic control task, the cognition-based framework for stabilization is established in this thesis.

It is assumed, that the mathematical model of the system to be controlled is unknown and fully controllable, as well as the state vector can be measured. The cognitive stabilizer is realized based on the cognitive framework by its four main modules: (1) “perception and interpretation” using system identifier for the system local dynamic online identification and multi-step-ahead prediction; (2) “expert knowledge” relating to the stability criterion to guarantee the stability of the considered motion of the controlled system; (3) “planning” to generate a suitable control input sequence according to certain cost functions; (4) “execution” to generate the optimal control input in a corresponding feedback form. Each module can be realized using different methods. In this thesis, “perception and interpretation” is realized using neural networks, Gaussian process regression, or combined identifier. “Expert knowledge” consists of the data-driven quadratic stability criterion, the quadratic Lyapunov stability criterion with a certain Lyapunov function, and the uniform stability criterion. The modules “planning” and “execution” are realized together with exhaustive grid search method or direct input optimization using inverse model. The whole cognitive stabilizer is realized using the autonomous communication among each module.

The cognitive stabilizer are tested using numerical examples or experimental results in this thesis. Pendulum system and Lorenz-system are considered as simulation

examples. Both are benchmark examples for the nonlinear dynamic control design. The cognitive stabilizer is experimentally implemented and evaluated to a three-tank-system. All the numerical examples and experimental results demonstrate the successful application of the proposed methods.

Kurzfassung

Das Thema dieser Arbeit ist ein kognitives Stabilisierungsverfahren, das basierend auf einem kognitionsbasierten Framework ein adaptives diskretes Regelungsverfahren darstellt. Ziel des kognitiven Stabilisierungsverfahrens ist es, eine spezifische Klasse von unbekannten, nichtlinearen, Mehrgrößensystemen autonom zu stabilisieren. Das kognitive Stabilisierungsverfahren ist in der Lage, relevante lokale Informationen über das unbekannte System zu erlangen. Es kann autonom geeignete Steuergrößen definieren, um das System zu stabilisieren.

Die Entwicklung von verschiedenen adaptiven, datenbasierten und modellfreien Reglern zeigte bereits die Tendenz der Erforschung von Regelungsmethoden mit hoher Autonomie. Der Begriff Autonomie wird hier verwendet, um die Tatsache zu beschreiben, dass das Regelungsverfahren bzw. die dazugehörige Programmierung so durchgeführt wird, dass der zugehörige Algorithmus den Rückführungsentwurf autonom ohne Einwirkungen von außerhalb des Algorithmus festlegen kann. Typische Methoden, die von dieser Definition beeinflusst werden sind die adaptive Regelungsmethode, die datenbasierte Regelungsmethode oder die modellfreie Regelungsmethode, deren Stand der Forschung in dieser Arbeit zusammengefasst wird. Der Hauptfokus liegt dabei auf der Autonomie der realisierten Verfahren. Es kann gezeigt werden, dass die existierenden Methoden immer noch einige offene Probleme aufweisen, um eine hohe autonome Regelung zu erreichen. Um diese offenen Probleme weiterzuentwickeln, kann ein Framework in den Ingenieurskontext eingeführt werden, das den Modellierungsverfahren bezüglich der menschlichen Kognitionsprozesse [Cac98] ähnelt und als kognitives Framework bezeichnet werden kann. Da Stabilisierungsaufgaben die elementarsten Regelungsaufgaben sind, wird in dieser Arbeit ein kognitionsbasiertes Framework zur Stabilisierung entwickelt.

Zunächst wird angenommen, dass das mathematische Modell des zu regelnden Systems unbekannt, vollständig steuerbar und der Zustandsvektor messbar ist. Der kognitive Stabilisierungsregler wird basierend auf dem kognitiven System durch seine vier Hauptmodule realisiert: (1) „Wahrnehmung und Interpretation“ durch einen Systemidentifikator zur Echtzeit-Identifikation der lokalen Systemdynamik und Mehrschritt-Vorhersage; (2) „Expertenwissen“ bezogen auf das Stabilitätskriterium um die Stabilität der betrachteten Bewegung des geregelten Systems zu garantieren; (3) Planung um eine geeignete Eingangsgrößensequenz nach bestimmten Gütefunktionen zu erzeugen; (4) „Ausführung“ um die optimalen Steuergrößen in eine entsprechende Rückführungsform zu generieren. Jedes Modul kann durch verschiedene Methoden realisiert werden. In dieser Arbeit wird das Modul „Wahrnehmung und Interpretation“ durch neuronale Netzwerke, Gauß-Prozess-Regression oder einen kombinierten Identifikator umgesetzt. Das Modul „Expertenwissen“ besteht aus dem datenbasierten quadratischen Stabilitätskriterium, dem quadratischen Lyapunov Stabilitätskriterium mit einer bestimmten Lyapunov-Funktion und dem gleichmäßigen Stabilitätskriterium. Die Module „Planung“ und „Ausführung“ werden zusammen durch

das inverse Modell mit dem vollständigen „Grid-Search“-Verfahren oder direkter Steuergrößenoptimierung realisiert. Die gesamte kognitive Stabilisierungsmethode wird durch die autonome Kommunikation zwischen jedem Modul realisiert.

Die kognitive Stabilisierungsmethode wird in dieser Arbeit durch numerische Beispiele oder experimentelle Ergebnisse getestet. Zwei Simulationsbeispiele (Pendel-System sowie Lorenz-System) werden betrachtet. Beide sind Benchmarkbeispiele für den nichtlinearen dynamischen Regelungsentwurf. Die kognitive Stabilisierungsmethode wird experimentell auf das Drei-Tank-System angewendet und die entsprechenden Ergebnisse werden bewertet. Die numerischen Beispiele sowie die experimentelle Umsetzung zeigen die erfolgreiche Anwendung des dargestellten Verfahrens.

Contents

Nomenclature	VIII
1 Introduction	1
1.1 State-of-the-art	1
1.1.1 Adaptive control techniques	1
1.1.2 Data-driven/Model-free control techniques	6
1.2 Tasks of the thesis	9
1.3 Organization of the thesis	10
2 Cognitive Stabilizer	11
2.1 Stabilization problem	11
2.2 Cognition-based framework for stabilization	12
3 Realization of the Cognitive Stabilizer	15
3.1 Realization of the module “perception and interpretation”	15
3.1.1 NARX-RNN	17
3.1.2 Gaussain process regression	24
3.1.3 Combined identifier	32
3.2 Realization of module “expert knowledge”	37
3.2.1 Data-driven quadratic stability criterion	39
3.2.2 Quadratic Lyapunov stability criterion with a certain Lya- punov function	41
3.2.3 Uniform stability criterion for switched nonlinear systems . . .	42
3.3 Realization of module “planning and execution”	43
3.3.1 Exhaustive grid search method	44
3.3.2 Inverse dynamic optimal control method	47
3.4 Realization of the whole framework	49

4	Simulation Results of Applications using Cognitive Stabilizer	53
4.1	Introduction of the simulation examples	53
4.1.1	Pendulum system	53
4.1.2	Lorenz-system	54
4.2	Simulation results	54
4.2.1	Simulation results of the pendulum system	55
4.2.2	Simulation results of the Lorenz-system	63
4.3	Summary	73
5	Experimental Application of Cognitive Stabilizer for a Three-Tank- System	74
5.1	Introduction of Three-Tank-System	74
5.2	User interface	76
5.3	Experiment results	77
5.4	Summary	82
6	Summary, Conclusion, and Outlook	84
6.1	Summary and conclusion	84
6.2	Outlook	86
	Literature	87

Abbreviations

ADP	Adaptive dynamic programming
AE	Average absolute error
ARE	Average absolute relative error
BIBO	Boundary input boundary output
BPTT	Back-propagation-through-time
DDPC	Data-driven predictive control
DLT	Dynamic linearization technique
GAP-RBF	Growing and pruning radial basis function
GPR	Gaussain process regression
I/O	Input/Output
IFT	Iterative feedback tuning
ILC	Iterative learning control
MIMO	Multi-input-multi-output
MISO	Multi-input-single-output
MFAC	Model-free adaptive control
MPC	Model predictive control
MRAC	Model reference adaptive control
MRAN	Minimal resource allocation network
MSE	Mean squared error
NARX	Nonlinear autoregressive exogenous model
NASC	Nonlinear adaptive switching control
PBSID	Predictor-based subspace identification
RBF	Radial basis function
RNN	Recurrent neural networks
RTPF	Real time particle filter
SIMO	Single-input-multi-output
SISO	Single-input-single-output
SVM	Support vector machine
3TS	Three-Tank-System
VRFT	Virtual reference feedback tuning

1 Introduction

In practice often the mathematical description of several systems can not be given easily or modeled accurate enough. Therefore classic control methods especially nonlinear ones are difficult to be applied, because the mathematical model of the system and the preliminaries of the system dynamic behavior are required to design a suitable controller. In order to overcome this problem, different kinds of adaptive, data-driven/model-free control methods are already developed for some classes of unknown systems. The development of such methods shows a clear tendency towards research on control methods with high autonomy. The control methods with high autonomy are the control methods without the requirement of prior mathematical model of the system, preliminaries of the system dynamic behavior, exact description of the changing environment of the system, and individual control design process. If the autonomous control can be realized, the whole control process can be robust, flexible, and efficient. As stabilization is the basic task of control problem, this thesis focuses on designing an autonomous stabilizer which has the following performances: 1) suitable for stabilizing unknown nonlinear dynamical MIMO systems; 2) can directly and optimally determine the control strategy by itself in the online process; 3) can stabilize the closed-loop system during the whole control process.

In order to discuss the motivation of this work more clearly, a state-of-the-art review about existing adaptive and data-driven/model-free control methods as well as a discussion about them will be given in Section 1.1. According to the advantages and disadvantages of the existing reviewed control methods, the detailed tasks of this thesis are defined in Section 1.2. Finally, the organization of this thesis will be given in Section 1.3. Some parts of this thesis are submitted as journal paper [NS15] or presented in international conferences [SZS12, MSS12, SS12, NS14].

1.1 State-of-the-art

The development of the two research areas adaptive control technique and data-driven or model-free control technique is reviewed separately in Subsections 1.1.1 and 1.1.2 in order to give an overview of the relevant current developments.

1.1.1 Adaptive control techniques

In the following, a brief introduction of adaptive control is given according to [LLMK11]. Adaptive control is a kind of technique which is able to adjust the controllers (to tune the controller parameters) automatically to achieve the control goals for the systems with unknown parameters. The corresponding scheme is shown in Figure 1.1.

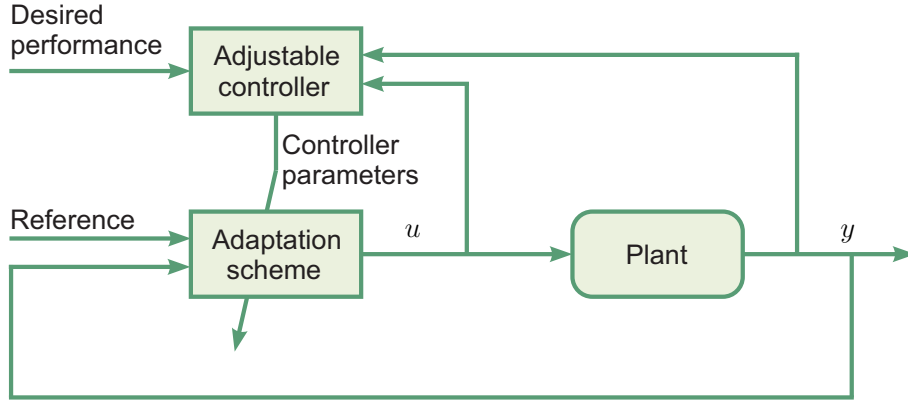


Figure 1.1: Scheme of an adaptive control system [LLMK11]

The adaptation scheme of adaptive control is realized using the measured inputs and outputs of the plant (which means that the unknown parameters of the plant are identified here) according to the desired performance. Using this adaptation scheme, the parameters of the controller, which can control the system outputs to the reference values, can be determined online. In essence, the adaptive control systems do not only reject the effect of disturbances acting upon the measured inputs and outputs, but also adjust the controller parameters automatically with disturbances acting upon the performance of the control system. A performance index to be defined manually using system variables is used to represent the performance of the controlled system. The performance index can be the input-output behavior of a transfer function or a tracking objective etc. A set of acceptable performance indexes are applied as the reference to modify the parameters of the controller.

The adaptive control techniques were developed at first with different algorithms such as self-tuning control, model reference adaptive control (MRAC), multimodel adaptive control etc. They are introduced separately in the following.

Self-tuning control was developed at first for controlling minimum phase discrete SISO dynamical system with known time delay and constant but unknown parameters in 1973 [AW73]. The optimal control strategy is designed with a fixed structure consisting of some predefined polynomials with estimated parameters. This strategy is suitable for applying pole placement method to achieve the desired control goal [AH80, BMG93]. Self-tuning control was improved later from 1991 with extended performance in order to make the controller more adaptive. For example, it can be used for MIMO systems with the same number of inputs and outputs as mentioned in [LLL91] where the MIMO system is considered as the combination of several interacting single-input-single-output (SISO) systems. With the help of an online model estimation using neural network, the self-tuning can also be used for unknown multi-input-single-output (MISO) systems with the measured input-

output data [ARC14]. As explained in [ARC14], the structure of the controller is designed according to the linearized estimated neural network model of the system using feedback linearization technique. The parameters of the controller are updated for each time step while the neural network is trained for each time step. This method can only be used for a certain class of MISO systems, because one weighting matrix of the neural network should be fixed in order to guarantee the performance of the controller. Additionally, the stability of the closed-loop system is only proofed analytically [ARC14].

The basic idea of MRAC [WYK58, KZSR13] is to design a linear reference model according to the desired dynamics of the closed-loop system and a suitable structure of an adaptive controller. Usually, the controller parameters are adjusted with an adaptation law with respect to the Lyapunov method using the error between the measured output of the nonlinear plant and the response of the reference model according to the desired output of the closed-loop system. Here, the reference model is a linear model and should be chosen with respect to the physical behavior or the mathematical model of the plant. As a result, a not suitable designed reference model can lead to an unsuccessful control result. For a nonlinear plant, the controller should be designed to cancel the nonlinearities in the plant in order to make the closed-loop system to follow the reference model.

Model reference adaptive control is also developed for unknown plants modeled using system identifier instead of known plants with an explicit mathematical description [LLMK11]. Furthermore, the controller structure can be determined using fuzzy rules [Koo01] while the adaptation law can be defined using other methods such as neural networks [LI95] etc. Despite these developments, MRAC is still not a perfect solution for the expectations of the high autonomy of the controlling of nonlinear MIMO systems, because the reference model and the controller structure can not be determined automatically especially for the variant plant or environment. Using MRAC, it is assumed that the plant model should always have stable zeros. Furthermore, the pole placement method is usually used to design the controller parameter. These lead to the difficulty of applying MRAC to MIMO systems [LLMK11].

Another kind of adaptive control technique is multimodel adaptive control [AC76, CA78]. This technique can be applied to linear MIMO systems with uncertainties. It is developed to solve the problem that the adaptive control can not always be used for real time applications. The reason is that for systems with large uncertainties of the unknown parameter vector, the parameter identification process is not fast enough for the real time applications [LLMK11]. Multiple model estimation algorithm is used for the online identification. Using this algorithm, a set of possible fixed models estimated using Kalman filters is established at first which is denoted as multi-estimator. The suitable model for control design is either a combination of multi-estimator with a weighting vector determined by the covariance matrix of state estimates or chosen among the multi-estimator with smallest estimation error according to a validation criterion. Therefore the computational requirement of

estimating a suitable model is reduced to only calculating the weighting vector or online validation of result. By combining the multi-estimator, there is no controller developed till now which can guarantee the stability of the closed-loop system. By choosing the best model with the smallest estimation error, multi-controller are designed according to the multi-estimator. This means that a suitable controller which is able to stabilize the closed-loop system and to realize the control goal is designed separately for each estimator of the multi-estimator [FAP06]. Therefore the global stability can not be guaranteed for this case. The most suitable estimator is determined online according to the current measurement of the system and a suitable switching logic. The difficulties using this control method are the design of the multi-estimator with enough possible estimators for every unknown parameter uncertainties and switching logic which should always be suitable for every desired control goal.

These three kinds of the adaptive control method are traditional adaptive control methods which can only be applied for the controlling of systems with unknown and time-varying parameters or for some unknown systems with strict constraints [LLMK11]. They are improved to solve the control problem with unknown disturbances for dynamics feedforward compensation, adaptive regulation etc.

Based on above mentioned traditional adaptive control methods, some adaptive nonlinear control methods (denoted also as nonlinear adaptive control methods) are developed extended specially for controlling some certain classes of nonlinear systems. For example using observer to estimate the uncertainties [CYW14], designing the controller with nonlinear H^∞ tracking performance [CC97], guaranteeing the stability of the closed-loop system with a certain Lyapunov function [GP08], and designing the controller with a certain form according to the fixed structure of the description of the system model [KI02, BKQ98] etc. A common shortage of these methods is the unavoidable requirement of a known structure of the system model.

In order to apply adaptive control to unknown nonaffine nonlinear systems (not only the systems with unknown parameters), adaptive fuzzy control [Lee10], adaptive inverse control [Zom94], adaptive neural network control [DLW08], model-free adaptive control (MFAC) [HJ11], and nonlinear adaptive switching control (NASC) method [CZW⁺11] are developed.

Adaptive fuzzy control applies the fuzzy logic for approximating the unknown nonlinearities of the system. Different control schemes were developed based on the Fuzzy description of the system model which includes the fuzzy-based H^∞ control scheme [Cha01], the dynamic surface control technique with fuzzy state observer [TLLL11], and the hybrid adaptive law [HG02] etc. which are mostly suitable for SISO systems. With respect to MIMO systems, only the systems with certain structure restrictions (for example an interconnection of several subsystems) are considered [Lee10, LTW07, CTL07, LTF10].

The basic idea of adaptive inverse control methods is to design a controller with the same model than that of the inverse model of the plant. Therefore suitable control inputs can be generated in accordance to a desired output or a desired reference model given manual [WW96]. For the case of unknown nonlinear systems, neural networks are usually used for identifying the model of the plant and for determining the corresponding adaptive algorithm of the controller [LFD05, Ple03]. The neural networks can also be used directly for identifying the inverse model of the plant [Zom94]. However, the stability of the closed-loop system was not considered using this kind of method.

Adaptive neural network control is developed with different algorithms. For example the feedback-linearization-based neural adaptive control [DLW08] is applied for unknown nonlinear SISO systems and is realized using a control law with fixed structure based on an affine-like input-output representation of the unknown system. This kind of representation can simulate the original nonaffine system and is trained online using neural network. It is assumed that the system to be controlled is Boundary-Input Boundary-Output (BIBO) stable and the second-order derivative of the nonlinear function of the unknown system with respect to the input exist. Using this control strategy, the BIBO stability of the closed-loop system can be guaranteed and the tracking control task can be realized.

Another kind of adaptive neural control is robust adaptive neural network control [CGH10, GW02] [TLZ11, DWW13] which is developed for controlling uncertain MIMO nonlinear systems with a certain known dynamical structure. Using this kind of control method, uncertain model of the system to be controlled is identified using neural networks and a control strategy can be realized using backstepping control [CGH10], variable structure control in combination with backstepping control [GW02], backstepping combined with decentralized control design principle [TLZ11], or the combination of a filtered tracking error with the implicit function theorem, input-to-state stability, and the small gain theorem [DWW13]. The semi-global uniform ultimate boundedness can be guaranteed and the tracking control goal can be achieved using such controller.

Model-free adaptive control (MFAC) [HJ11] approach was developed as an extension of adaptive control of tracking tasks for a class of discrete time MIMO nonlinear systems which have the same number of inputs and outputs. Using MFAC, the known dynamical structure of the nonlinear system model is not required, even the identification of the system model is not needed. A series of linearized local dynamic models of the closed-loop system along the dynamic operation points is established with online input-output (I/O) measurements without training process. A suitable linearization structure is chosen directly among different possibilities of dynamic linearization technique (DLT) with pseudo-partial derivative (PPD). Each possibility uses different PPD matrix to describe the system model. All kinds of PPD matrix should be a diagonally dominant matrix. In order to guarantee

this condition, the given I/O data should include enough and accurate information about the system dynamic behavior which needs additional afford to be guaranteed in practice. Each possibility of DLT leads to its own controller structure, whose parameters are determined according to the linearized data model updated using the online I/O measurement. The MFAC algorithms allows BIBO-related statements about the stability of the closed-loop system.

Nonlinear adaptive switching control (NASC) method [CZW⁺11] is another kind of adaptive model-free control method for a class of nonlinear MIMO systems with the same number of inputs and outputs. Using this method, the unknown system is described as the sum of two parts: controller-driven model and virtual unmodeled dynamic. The controller-driven model is a low order linear model defined prespecified by designers and can be used for controller design with some off-line trial for the determination of the parameters of the controller. The virtual unmodeled dynamic is determined using adaptive-network-based fuzzy inference system based estimator in order to describe the unmodeled difference between the system to be controlled and the controller-driven model. With the controller-driven model and the virtual unmodeled dynamic, two suitable controllers are separately designed according to the desired trajectory of the closed-loop system and a certain cost function. These two controllers are selected to be applied to the system to be controlled according to a suitable switching algorithm.

As shown in the brief introduction of the different adaptive control methods, MFAC and NASC can partially satisfy the data-driven requirements of the high autonomy for the controller design of MIMO systems in the context of stabilization. As discussed above, there are still strict constraints by applying them.

1.1.2 Data-driven/Model-free control techniques

Except the adaptive control techniques, several controller design methods denoted as data-driven or model-free control technique have already been developed trying to achieve the expectations of the high autonomy for controlling nonlinear systems, like virtual reference feedback tuning (VRFT) [CS06], iterative learning control (ILC) [XT03], iterative feedback tuning (IFT) [HGGL98], real time particle filter (RTPF) [KFM04], data-driven predictive control (DDPC) [KSZ09], and adaptive dynamic programming (ADP) [MCLS02] etc. Brief introductions to these methods are given in this subsection.

In VRFT approach [CS06], an optimal controller is determined among a special controller class according to the collected data of the considered nonlinear unknown SISO system. The VRFT approach can also be extended for MIMO systems, however, according to [RFAJV12], is restricted to the same number of the inputs and outputs. The process to determine the controller is realized without identification of the system model off-line, so the system should be a time invariant one. The given

data should include enough information of the systems dynamic behavior in order to guarantee the optimal controller working as desired for the real system. Additionally, it is assumed that the map of the nonlinear system is smooth and invertible. Such requirements are usually not easy to be achieved in practice due to the assumed complexity of unknown systems. Moreover, the stability of the controlled system is not discussed within this approach.

The main idea of the ILC approach [XQ98] is to update the desired control input according to the output measurements iteratively for each discrete time step using always the same initial conditions of the system dynamic behavior for each iteration, which is an imperative fundamental assumption of ILC [XT03,ZL15]. A priori knowledge about the boundaries of the system dynamic behavior is required for the learning process [XQ98], so ILC is not a complete model-free approach. On the one hand, the perfect tracking (as advantage) of the desired system outputs can be realized using ILC without any identification of the system model. On the other hand, ILC can not be used online because the outputs can not always be measured within the same initial conditions of the system states and the same control environment within one discrete time step in practice. The ILC is combined with identification method to achieve the online application as shown in [BKdJS05]. However, the advantage of the perfect tracking is missing because of the not avoidable identification error.

The IFT approach [HGGL98] is a model-free control method without included identification process. The structure of the controller is predefined. The parameters of the controller are optimized for each time step using 3 experiments: collection of the output signal corresponding to some input signal, calculation of the gradient of the controller parameter with the help of the Gauss-Newton algorithm, and determination of the most optimal according to a certain criterion. Therefore this approach is also difficult to be applied in real time. The IFT was also developed for nonlinear systems with some assumptions [SBA⁺03]. The stability guaranteed here is the BIBO stability under a certain condition that the initial conditions of the experiments almost not affect the closed-loop response [SBA⁺03]. Moreover, it is only suitable for MIMO systems under some strict conditions [GCR03].

The RTPF [KFM04] is usually applied for system dynamics estimation and robot localization problems. The dynamic behavior of the unknown system can be represented and predicted as the posterior probability densities of the measured system samples, which is sampled according to the inputs and sensor data collected by the dynamical system/robot moving around a given map of possible outputs. If the sampling of the required samples of an unknown nonlinear dynamical system can be achieved successfully, the suitable inputs for the desired outputs can be determined from the predictive densities directly. A given map for a dynamical system is not always available. Although with a given map a sampling process which includes enough information of the system dynamics is also expensive for online process.

Furthermore, the stability of the controlled system is also not considered within this approach.

Model predictive control (MPC) was developed at first for discrete time linear systems and extended to the stabilization and tracking control problem for nonlinear systems [LM67]. With a known function of the system model, the future system dynamic behavior can be predicted, stabilized, and optimized according to some certain criteria using different kinds of methods [GP11]. In order to apply MPC to unknown systems, data driven predictive control (DDPC) [FM98, KHR03] was developed with subspace identification. However, it can only be used for controlling linear systems. With the help of neural networks identifying the unknown system, DDPC has been extended to control nonlinear systems, but requires more or less the preliminaries of the system model as published in [KSZ09, WSL13] or has some constraints such as only suitable for SISO systems as shown in [QLS15].

The ADP approach [MCLS02] was developed for the optimal control problem of nonlinear dynamical systems which usually requires to solve the nonlinear Hamilton-Jacobi-Bellman (HJB) equation. However, HJB can not be exactly solved for general nonlinear MIMO systems. In order to solve this problem, the ADP algorithm was developed with three neural networks (the model network, the critic network, and the action network). The model network is used for model identification, the critic network is applied for updating the optimal cost function related to the control law, and the action network is used for updating the control law [LWZ11]. The computational requirements are large using ADP because of the use of three neural networks, of which the approximating accuracy should be guaranteed to achieve the control goals. Additionally, the stability of the closed-loop system is not considered in the controller design. In order to solve this problem, robust ADP was developed [JJ14] for controlling nonlinear systems with dynamic uncertainties. The system model is assumed to consist of two parts: unknown system dynamics with measured states and unknown system dynamics with unmeasured states. With the help of an iterative technique called policy iteration and nonlinear small-gain theorem, both parts are considered during the controller design to stabilize the closed-loop system at the origin. However, this method can only be used for SIMO systems. As an extension of ADP, robust ADP [ZCZL11] was developed for optimal tracking control problem. The assumption of the known system order can be solved using robust ADP with the help of recurrent neural network as the model network. However, in order to achieve the asymptotical tracking results, some other assumptions about the boundary of the identified model are needed.

From the introduction and discussion of the existing methods, it can be seen that none of these methods can realize the expectations without any shortage related to the high autonomy. For example, some of the existing controllers can only be used for SISO or for MIMO systems under some strict conditions like the number of the inputs and outputs of the system should be same. Some of the controllers

can be used for general MIMO systems. However, they need some preliminaries of the nonlinearities of the system to be controlled or the stability of the closed-loop system is not considered by the controller design.

As stated before, the control method with high autonomy should be suitable for the most kinds of unknown linear and nonlinear MIMO systems without any strict condition and preliminaries about the system dynamic behavior. Otherwise, manual design of a suitable controller is always necessary for different systems. In the context of the stabilization control methods, the stability of the closed-loop system should be considered in the whole design process. To be suitable for general cases, the stability considered here can be input/output stability, BIBO stability, or state stability. In other words, the controller should be designed automatically to guarantee the stability of the closed-loop system and to stabilize the system with the desired dynamic. It can be concluded from the state-of-the-art, that there is still no method which can satisfy the expectations of the high autonomy without any shortage. Therefore a new kind of controller for stabilization task should be designed in order to overcome the shortcomings or give an alternative of the existing methods.

1.2 Tasks of the thesis

In this thesis, the high autonomy of a stabilizer is desired to realize the following properties:

- It should be suitable for stabilizing unknown nonlinear dynamical MIMO discrete systems.
- The dynamic behavior of the concerned unknown system has to be learned by the desired algorithm only with the knowledge about the I/O measurements. It should be mentioned that in most practical cases the learned system dynamic behavior describes only the local dynamics of the system to be controlled accurately, which means it cannot be globally accurate. Therefore the desired stabilizer should be able to learn the system dynamic behavior online, in order to update the actual dynamic behavior. With this capability, the dynamics of time variant systems can also be learned online.
- Using the learned system dynamic behavior, the desired stabilizer should be able to do multi-step prediction of the system outputs. With this prediction, the desired control inputs for the upcoming time steps should be directly, optimally, and automatically determined according to the learned system dynamic behavior.

- The stability of the closed-loop system should be guaranteed in the whole control process. The stability here is denoted as the stability of the upcoming motion related to the local dynamic behavior. Furthermore, the required stability criteria should be realized in a data-driven manner, because the system model is not known.

Such kind of ability matches the capability of cognition which was introduced into control technique using a cognition-based framework [AS08]. The idea of designing the stabilizer using cognition-based framework with respect to the requirement of the high autonomy should first be explained. Suitable algorithms should be designed to realize the cognition-based framework for the stabilization task. If different realization possibilities exist, an automatic selection strategy among them should be designed in order to apply the most suitable one to the considered system. Finally, the possible realizations should be applied to some simulation and experiment examples in order to evaluate the performance of them.

1.3 Organization of the thesis

According to the tasks, this thesis is organized as follows:

- In Chapter 2, the concept of the cognition-based framework for stabilization is introduced. The problem to be solved is first defined in detail in Section 2.1. The cognitive-based framework is established and explained in Section 2.2.
- In Chapter 3, the realizations of cognition-based framework is first presented separately for each module of the framework in Sections 3.1, 3.2, and 3.3. In Section 3.4, the realization of the whole framework is discussed.
- In Chapter 4, the cognitive stabilizer is applied to two simulation examples in order to show the performance of it. They are introduced with the mathematical description in Section 4.1. The corresponding simulation results using different realizations of the whole cognitive framework are discussed in Section 4.2.
- In Chapter 5, the cognitive stabilizer is applied to a real three-tank-system. In detail, the system is introduced, the user interface is established as well as the experimental result is shown and analyzed.
- In Chapter 6, summary and conclusion as well as outlook of the proposed cognitive stabilizer are outlined.

2 Cognitive Stabilizer

As mentioned in Chapter 1, in the context of data-driven adaptive stabilizer for nonlinear system, if a precise description of the concerned system is not known, it has to be identified (learned) at first by the controller using the measured input/output data. In order to overcome the shortcomings of the possible inaccuracy identified model, the required stability criteria of the controller should also be realized in a data-driven manner instead of a model-based stability criterion. The control strategy should be generated automatically with respect to a suitable optimization criterion using the knowledge from both, the identified model and the stability criterion online.

Such kind of ability matches the capability of cognition which is a distinctive feature of cognitive systems [VMS07]: “being able to understand how things might possibly be, not just now but at some future time, and to take this into consideration when determining how to act”. Therefore cognition described by a cognition-based framework was introduced into system control to solve the high autonomous control problem [Cac98, AS08]. Due to the focus of this thesis, the cognition-based framework is further developed for the stabilization problem.

In order to explain the cognition-based framework explicit, the stabilization problem to be solved in this thesis is first defined in Section 2.1. The detailed description and requirement of the cognition-based framework is stated in Section 2.2. This chapter is presented based on [NS14, NS15].

2.1 Stabilization problem

Different definitions of stabilization in the control research area are known. It is necessary to define the stabilization problem which should be solved in the context of cognitive stabilizer clearly.

The considered class of time variant nonlinear MIMO systems is given in the discrete form by

$$x(k+1) = f(x(k), u(k), k), \quad (2.1)$$

where $f(\cdot)$ denotes the nonlinear system discrete dynamic, $x(k) \in \mathbb{R}^n$ the state vector, $u(k) \in \mathbb{R}^m$ the control input, and k the current discrete time step. The stabilization problem should be solved by designing a suitable control input vector

$$u(k) = g(x(k), x(k-1), \dots, x(k-l), k) \quad (2.2)$$

with $g(\cdot)$ as the function of the control input and l as an integer which is smaller than k . With the suitable control input, the system to be controlled can be stabilized at an

arbitrary given physical possible state x_e . There are different definitions of stability: Lyapunov stability, input/output stability, BIBO stability etc. The suitable stability criteria should be realizable using the cognition-based framework which is detailed in Section 3.2. Furthermore, it is assumed that the function $f(\cdot)$ is unknown, the state vector x can be measured, and the system is fully controllable.

2.2 Cognition-based framework for stabilization

The term cognition origins from psychology and is defined in [Nei76] as “the mental activity of knowing: the acquisition, organization, and use of knowledge”. A typical human cognitive process [Cac98] includes four cognitive functions such as perception, interpretation, planning, and execution as well as two cognitive processes such as allocation of resources and memory/knowledge base.

Cognition in the engineering context, in order to be distinguishable from the definition of cognition in psychology, means the capability of perceiving the environment, assimilating information by discovering and structuring knowledge, and autonomously making rational decisions of how to act [SHH⁺95, AS08]. Similar to the human cognitive process, the four cognitive functions and two cognitive processes describe the cognition behavior for a high autonomous control process. This can be clearly illustrated by a framework as developed in [Cac98]. Based on the cognitive framework for system control, a cognition-based framework in the context of stabilization control task is developed as shown in Figure 2.1.

In this thesis, the controller based on such framework used for stabilization task is denoted as cognitive stabilizer which is designed to realize all the expectations stated in Section 1.2.

As shown in Figure 2.1, the framework consists of two main parts: external environment and cognitive stabilizer. In the external environment, the mathematical model and physical behavior of the system to be stabilized are not known. The inputs of the system are provided by the actuators and the outputs which are also assumed to be the states of the system are measured using the sensors. The disturbances coming from the environment act possibly on the actuators, the plant or the sensors. The number and the function or the values of the disturbances are also unknown. The inputs and outputs including the disturbances are the only information about the unknown system which are transmitted to the cognitive stabilizer. Furthermore, the user as a part of the external environment defines an explicit goal of the stabilization task such as at which state should the controlled system be stabilized (which value should x_e have) or which kind of stability criterion should be considered or how should be the control properties (e.g. low energy requirement, short time to reach the goal dynamic etc.). Logically, this goal should also be transmitted to the stabilizer in order to let the stabilizer know what should it do.

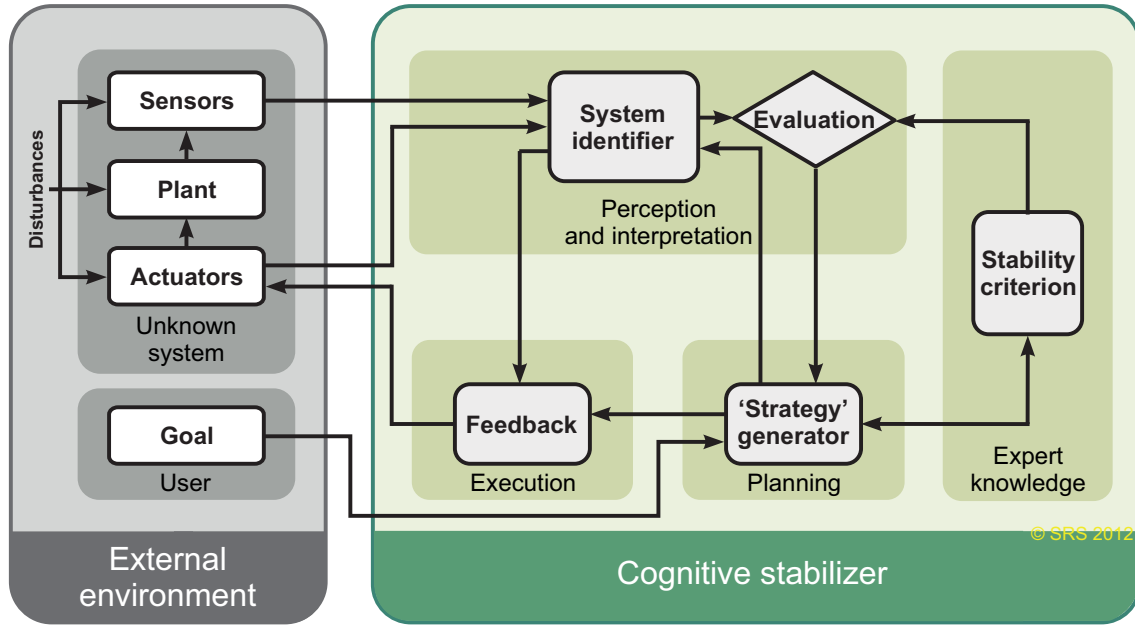


Figure 2.1: Cognition-based framework for stabilization [NS15]

The cognitive stabilizer consists of four modules (“perception and interpretation”, “expert knowledge”, “planning” as well as “execution”) and their interconnections.

In the module “perception and interpretation”, the local dynamic of the unknown system should be learned autonomously for analyzing and predicting the system dynamics in order to design a suitable stabilizer for this system later in the module “planning”. Therefore, a training process is necessary for learning the current dynamic of the system. In detail, a mathematical description of the system input-output behavior should be established by system identifier using only the inputs and outputs measured from the actuators and sensors. Using the learned mathematical description, it should be able to predict the system dynamics in the future for a certain multi-step prediction horizon with the horizon size s accurately with certain test control inputs. According to the knowledge about the stability from the module “expert knowledge”, the stability of the predicted system dynamics can be analyzed and evaluated.

In the module “expert knowledge”, different kinds of stability criteria are stored here. They provide the theoretical support for achieving different stabilization tasks for different systems. The suitable stability criterion to be used should be decided autonomously in the module “planning” according to the control goal given by the user. The selected stability criterion provides a condition for achieving a stable motion of the system in the future. This condition can be used directly for determining the suitable control inputs to be updated or indirectly as an evaluation according

to the judgments of the stability of the motion of the predicted system using the possible inputs in the module “planning”.

In the module “planning”, the strategy for choosing the most suitable control input should be determined automatically. At first, a range of possible inputs for the system in the upcoming prediction horizon should be generated according to the limitation of the physical meaning of the system inputs. Secondly, depending on the control goal defined by the users, the suitable stability criterion is chosen and decided to be used either directly or indirectly. It can be used directly as a rule of determining the new control input which can lead a stable behavior of the system in the upcoming steps. It can also be used indirectly with the predicted system dynamics in the module of “perception and interpretation” together. For example, the stability of the predicted system dynamics with a certain input can be analyzed and evaluated according to the stability criterion and the evaluation result can be used to judge whether the considered input value is suitable for the stabilization task. Therefore, a strategy for determining the suitable inputs among all possible inputs can be decided. Finally, in order to choose the most suitable control input, a suitable cost function is defined according to the control goal giving by the users and the most suitable control input is determined with respect to this function autonomously.

According to the determined strategy in the module “planning”, the most suitable control input is generated in the module “execution” in the form of feedback control with the predicted system outputs using the identified mathematical description of the system and be given to the actuators of the unknown system.

The process discussed above represents one complete interaction between the cognitive controller and the unknown system. Every module in the framework serves its specific functionality and can be realized by applying different methods. However, the realization of each module must be chosen in consistent and proper way such that no communication problems among different modules can be encountered. Here it is assumed that no further problems appear. Possible realization method of each module and the whole framework is given in the next chapter.

3 Realization of the Cognitive Stabilizer

As mentioned, different methods can be used to realize the modules in the framework of the cognitive stabilizer. In this chapter, possible methods for each module are discussed. They are given first separately (Section 3.1 for the module “perception and interpretation”, Section 3.2 for the module “expert knowledge”, and Section 3.3 for both modules “planning” and “execution”) in detail. The realization possibilities of the whole framework by autonomous connection of each module is delineated in Section 3.4. Some parts of this chapter are published in [SS12, NS14] or submitted in [NS15].

3.1 Realization of the module “perception and interpretation”

Various kinds of methods can be used to identify and predict the system dynamics of unknown nonlinear MIMO systems with the I/O measurements online, such as recurrent neural networks (RNN) [Hay99], Gaussain process regression (GPR) [RW06], adaptive radial basis function (RBF) neural networks [SK10], support vector machine (SVM) [SS09], Predictor-based subspace identification (PBSID) [Chi07], real time particle filter (RTPF) [KFM04] etc. In the following, they are introduced briefly.

Recurrent neural networks have one or more feedback loops between the neurons and can be used for identifying unknown nonlinear MIMO systems based on different network structures [Hay99]. One of the structures is the nonlinear autoregressive with exogenous (NARX) [Hay99, PRA00]. As stated in [SHG97], different from other recurrent networks consist of more than one feedback coming from the output neuron and hidden states to the input neuron, NARX networks have a limited feedback between the output neuron and the input neuron. This results a less computational load and the unknown systems can be identified faster than using other kind of RNN [SHG97]. Additionally, NARX-RNN can be used for multi-step-ahead prediction of unknown nonlinear systems with iterative prediction process [PRA00].

Gaussain process regression [RW06] is based on a statistic model which does not learn the model of the considered unknown nonlinear MISO system by adjusting the parameters of a given assumed fix structure, but by finding the probability distribution of all possible functions related to the system. Additionally, GPR can be applied directly for the multi-step-ahead prediction of unknown systems for a certain prediction horizon without iterative prediction process [GRMS03]. These two points result to a less computational load of the identification and prediction process. However, the covariance function, which represents the statistical model of

GPR, is usually determined experimentally [KGBMS05]. Moreover, for identifying MIMO systems, GPR should be used repeatedly for each system output.

Adaptive RBF neural network [SK10] can also be used for the identification of nonlinear time-varying dynamical MIMO systems only with the measured I/O measurements. There are two different kinds of adaptive RBF neural network: growing and pruning radial basis function (GAP-RBF) as well as minimal resource allocation network (MRAN). Different from other network architectures, the dimension of the hidden neurons of them is not predetermined but adjusted automatically according to the complexity of the system to be identified. However, this characteristic leads a high computational complexity of the adaptive RBF neural network. Using MRAN, all neuron parameters should be updated during each learning process. Using GAP-RBF, although some of the neuron parameters should be updated, the computational complexity is still high because of the high sensitivity of the network initial threshold values to be determined using a large number of the neurons.

Support vector machine [SS09] is usually applied for both linear and nonlinear time series prediction. The principle idea of SVM is to find a suitable mathematical description consisted of several possible functions with trained optimal weights and threshold. The possible functions belong to a same class of functions which is denoted as kernel function. Similar to GPR, there is also no predefined structure of the description of the system, which means that SVM is a data-driven prediction approach. This approach is usually applied for SISO systems and can be extended to MIMO systems [SFdPCAGPC04]. However, as stated in [SS09], no algorithm for choosing the optimal kernel function is known to be fast enough to handle the computational load in real time.

Predictor-based subspace identification is developed firstly for identification of linear dynamical MIMO systems [Chi07] and extended for Hammerstein-Wiener nonlinear MIMO systems [GB10]. Using PBSID, a state space representation of the system with unknown system matrices is established at first and the unknown matrices are estimated using a certain formulation of least square problem [Chi07]. This results a simple controller design process, because a lot of control methods based on the state space representation can be applied directly. However, the typical nonlinear system is not of those class denoted by the Hammerstein-Wiener nonlinear model class and can not be represented as a series consisted of input nonlinearity, linear model, and output nonlinearity. Therefore PBSID is difficult to be applied to a complete unknown nonlinear systems.

Real time particle filter [KFM04] is a sample-based variant of Bayes filters, which is usually applied for the robot localization. It can also be used for the identification of unknown nonlinear systems. Using RTPF, enough input and output measurements denoted as observations are collected at first. The observations are considered to calculate the distribution of the sample sets within a certain update window. The most important sample sets are selected by determining the sample weightings.

With the most important sample sets, the system can be identified. This kind of identification method needs a large amount of the input and output measurements which can not always be provided in the practice.

It can be concluded from the brief introduction above, each of these methods has its own advantages and disadvantages. In order to achieve the identification task for most kinds of systems with less shortage, a combined identifier should be designed based on the existing identification methods in the module “perception and interpretation” of the cognitive framework. It is desired, that the cognitive stabilizer can decide according to the I/O measurement which kind of system identifier is suitable for the identification of the system to be controlled. It should be mentioned here, for the same system to be controlled under different conditions or with different training I/O measurement, the suitable identifier may be not the same one. Therefore, the tuning algorithm in the combined identifier should also be designed online.

In this thesis, NARX-RNN and GPR are choosing to realize the module “perception and interpretation”. Other methods will be discussed in the future work. In the following subsections, brief introductions and performance evaluated using simulation results for both methods (NARX-RNN and GPR) are given. Additionally, a combined identifier based on both methods is developed with an autonomous tuning algorithm in order to realize the module “perception and interpretation” more adaptive and accurate.

3.1.1 NARX-RNN

Using recurrent neural networks (RNN), dynamical system behavior can be learned without a high level memory also for high dimensional nonlinear systems [Hay99]. Nonlinear autoregressive exogenous model as a kind of RNN can be applied not only for the system dynamic behavior learning but also for multi-step-ahead prediction of the system states according to new system inputs with high accuracy [SHG97]. A brief introduction of NARX-RNN is given in the following.

Same as all other kinds of neural network, the prediction of system outputs using NARX-RNN consists of two steps: training phase and prediction phase. During the training phase, some measured input and output training data from the real or simulated system are used to train the weighting matrices and bias of the neural network, which are used to build the mathematical description of the unknown system. In the following, k is denoted as the current discrete time step, m as the number of the system inputs, r as the number of the system outputs, $u(k) \in \mathbb{R}^{m \times 1}$ as the input vector, and $y(k) \in \mathbb{R}^{r \times 1}$ as the output vector of the system at k . In order to represent the system dynamic behavior, exogenous input vector u_N and delayed output vector y_N defined as

$$\begin{aligned} u_N(k-1) &= [u(k-1), \dots, u(k-l-1)] \text{ and} \\ y_N(k-1) &= [y(k-1), \dots, y(k-l-1)] \end{aligned} \quad (3.1)$$

with l denoted as the lag (the size of the training horizon) are given as the input vectors of the NARX-RNN in the training phase. The output vector of the NARX-RNN is the actual output vector of the system $y(k)$. The structure of NARX-RNN with the corresponding input vectors and output vector is illustrated in Figure 3.1.

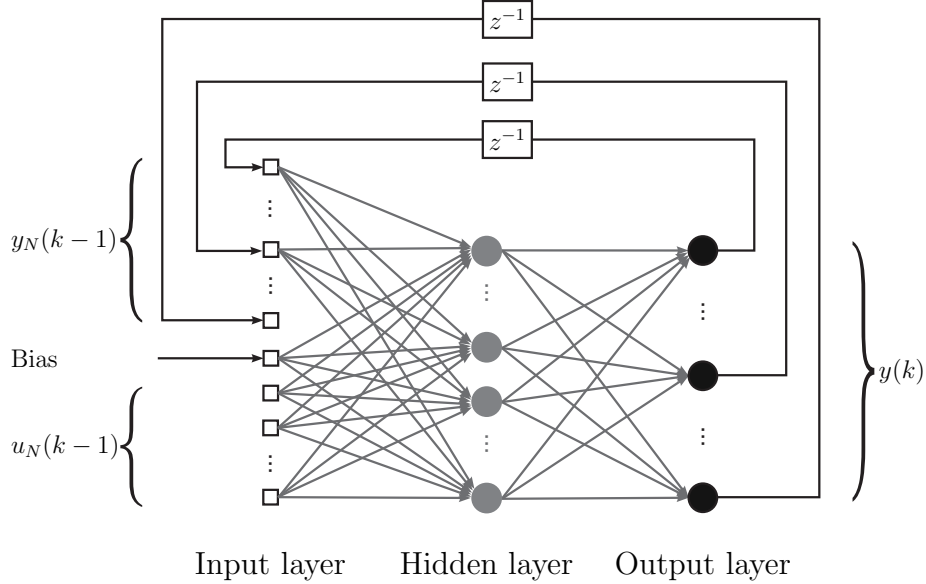


Figure 3.1: Structure of NARX-RNN [Hay99]

The network consists of three distinct layers: an input layer, a nonlinear hidden layer with the active function f_1 , and a linear output layer with the active function f_2 . The number of the neurons η in the hidden layer is predefined, which is discussed later. The structure of the relation between the output vector and the input vectors is predefined by NARX-RNN as

$$y(k) = f_2(W_3 \cdot f_1(W_1 \cdot u_N(k-1) + W_2 \cdot y_N(k-1) + b_1) + b_2) \quad (3.2)$$

with W_1 , W_2 , and W_3 as weighting matrices, b_1 and b_2 as bias vectors. This can be represented functionally more clearly in Figure 3.2

The weighting matrices and bias are trained here using back-propagation-through-time (BPTT) algorithm to enable the NARX-RNN to approximate the measured system inputs and outputs with high accuracy.

During the prediction phase, the new system output vector for the next step $\hat{y}(k+1)$ can be predicted using the trained weighting matrices and bias with a new system

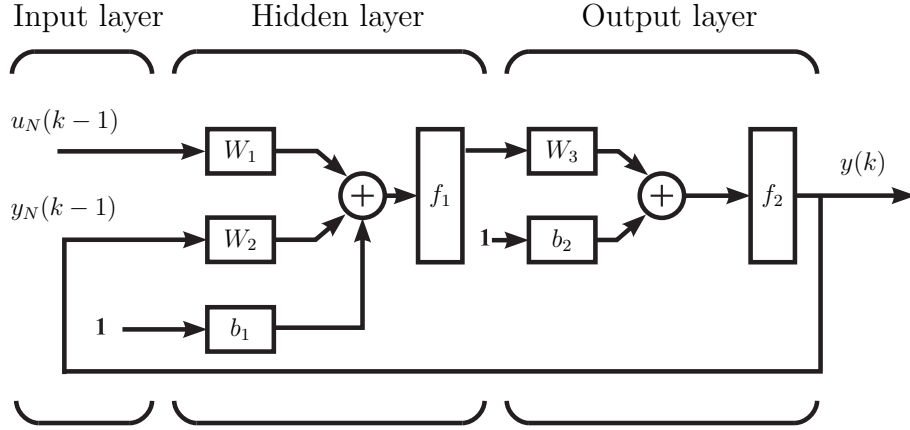


Figure 3.2: Functional structure of NARX-RNN

input vector $u_d(k)$ as

$$\hat{y}(k+1) = f_2(W_3 \cdot f_1(W_1 \begin{bmatrix} u_d(k) \\ u(k-1) \\ \vdots \\ u(k-l) \end{bmatrix} + W_2 \begin{bmatrix} y(k) \\ y(k-1) \\ \vdots \\ y(k-l) \end{bmatrix} + b_1) + b_2). \quad (3.3)$$

Iteratively, the system outputs in the multi-step prediction horizon with the horizon size s can be predicted with the real system outputs from the training horizon and the previous predicted system outputs as

$$\begin{aligned} \hat{y}(k+i) = & f_2(W_3 \cdot f_1(W_1 \begin{bmatrix} u_d(k+i-1) \\ u_d(k+i-2) \\ \vdots \\ u_d(k) \\ u(k-1) \\ \vdots \\ u(k+i-l-1) \end{bmatrix} + W_2 \begin{bmatrix} \hat{y}(k+i-1) \\ \hat{y}(k+i-2) \\ \vdots \\ \hat{y}(k) \\ y(k-1) \\ \vdots \\ y(k+i-l-1) \end{bmatrix} \\ & + b_1) + b_2) \end{aligned} \quad (3.4)$$

with $i \in [1 \dots s]$.

Using NARX-RNN to learn the system dynamic behavior for every multi-step prediction horizon, the outputs of the unknown nonlinear system can be predicted online according to the new system inputs.

It is necessary to mention several points related to application of NARX-RNN for learning and prediction the system dynamic behavior. In order to state these points clearly, simulation examples using NARX-RNN for benchmark nonlinear chaotic system (Lorenz-system) described by

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{pmatrix} = \begin{pmatrix} -\sigma y_1 + \sigma y_2 + u_1 \\ -y_1 y_3 - y_2 + r y_1 + u_2 \\ y_1 y_2 - b y_3 \end{pmatrix}, \quad x(t=0) = x_0 \quad (3.5)$$

with $\sigma = 10$, $r = 28$ and $b = \frac{8}{3}$ is discussed in this chapter.

- From the structure of the NARX-RNN, it is clear that NARX-RRNN can be directly used for the identification and prediction of MIMO systems.
- During the training phase, the approximation results are validated to guarantee the suitability of the weighting matrices and bias to approximate the system dynamics with high accuracy. In the prediction phase, no validation to check the accuracy of the predicted outputs is carried out. Therefore, a simple but efficient additional validation for the predicted outputs is defined here. Assume $\Delta y_{max,q} = \max |y_q(i+s) - y_q(i)|$ with $q = 1 \dots r$ and $i = 1 \dots k$, the predicted system outputs are considered as accurate enough (correct) if the condition

$$|\hat{y}_q(k+s) - \hat{y}_q(k)| < \lambda \Delta y_{max,q} \quad (3.6)$$

is fulfilled, where λ is a predefined tolerance parameter. The idea of this kind of validation can be explained from a practical point of view. Each system has its own dynamic and energy limit of the output side generated from the input side. Therefore the predicted change of the system outputs in a time period with a certain length can normally not be much larger than any past change within any time period with the same length.

- If the predicted system outputs are not validated as correct, the learning phase of the NARX-RNN should be repeated with an increased number of the neurons in the hidden layer until the validation result is positive. With this strategy, the determination of the parameter η is also achieved autonomously by the neural network.
- Using NARX-RNN, a suitable value of prediction horizon size s should be determined experimentally for different systems for a given acceptable prediction accuracy. For example, assume the sample time T_s is taken as 10^{-3} s, the training time $T_t = 0.5$ s, $\eta = 30$, $\alpha = 0$, and the initial condition of the outputs $x_0 = [-10 \ 10 \ 25]$, the suitable value of s in this case is 5 by applying NARX-RNN for identifying the Lorenz-system (3.5) with a spline input training data

and predicting the outputs of the Lorenz-system with spline input test data. For $s > 5$ the prediction accuracy can not be guaranteed. Repeat the identification and prediction phase for 0.1 second, the corresponding training phase and the prediction result are shown in Figure 3.3.

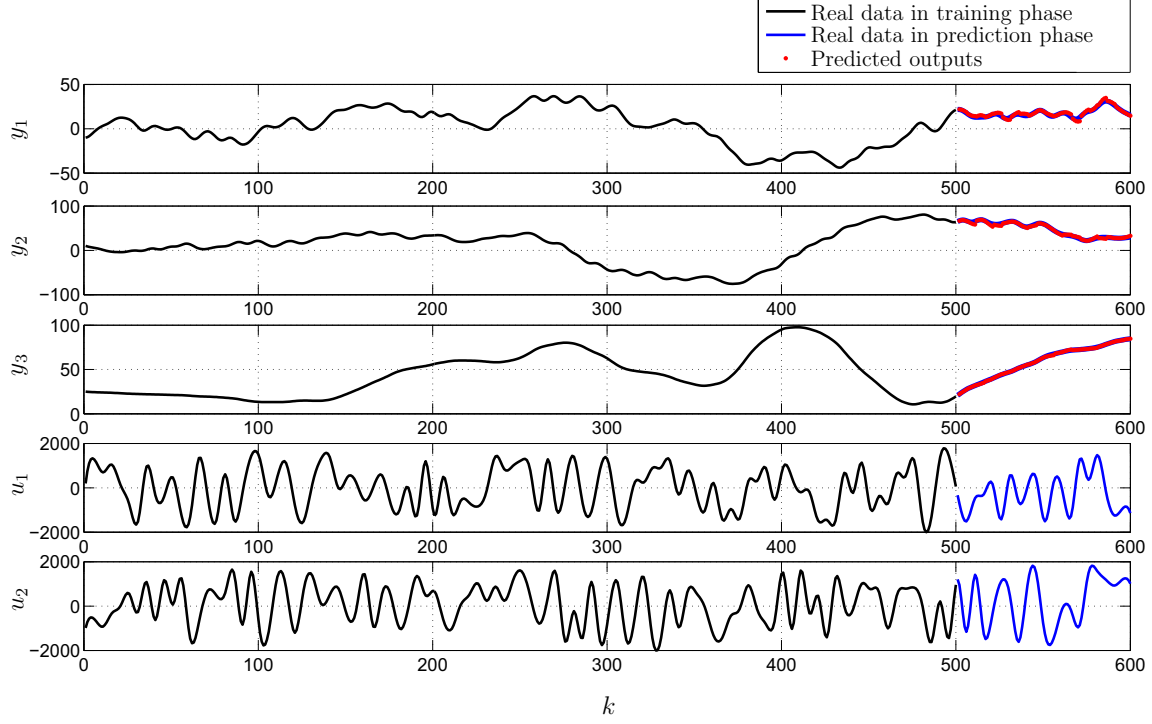


Figure 3.3: Comparison of real and predicted outputs using NARX-RNN with spline input training and test data

The two input and three output data in the training phase (the first 500 steps) are denoted with black lines. The blue lines show the test data of the inputs and outputs in the prediction phase. The predicted outputs denoted by the red points clearly indicate that NARX-RNN with spline input training data can be used for predicting the outputs of nonlinear dynamic MIMO systems with acceptable accuracy and suitable value of s .

- During the training phase, the input training data can be generated in different way, such as standard uniform distributed random signals, standard normal distributed random signals, step functions, and spline functions ect. In order to decide the type of the training input data in practice, it is necessary to check the prediction performance of the NARX-RNN with different typical training input data.

With the same assumption, the Lorenz-system (3.5) is identified and predicted with different training and test input data. The corresponding simulation

results for the prediction phase are shown in Figure 3.4.

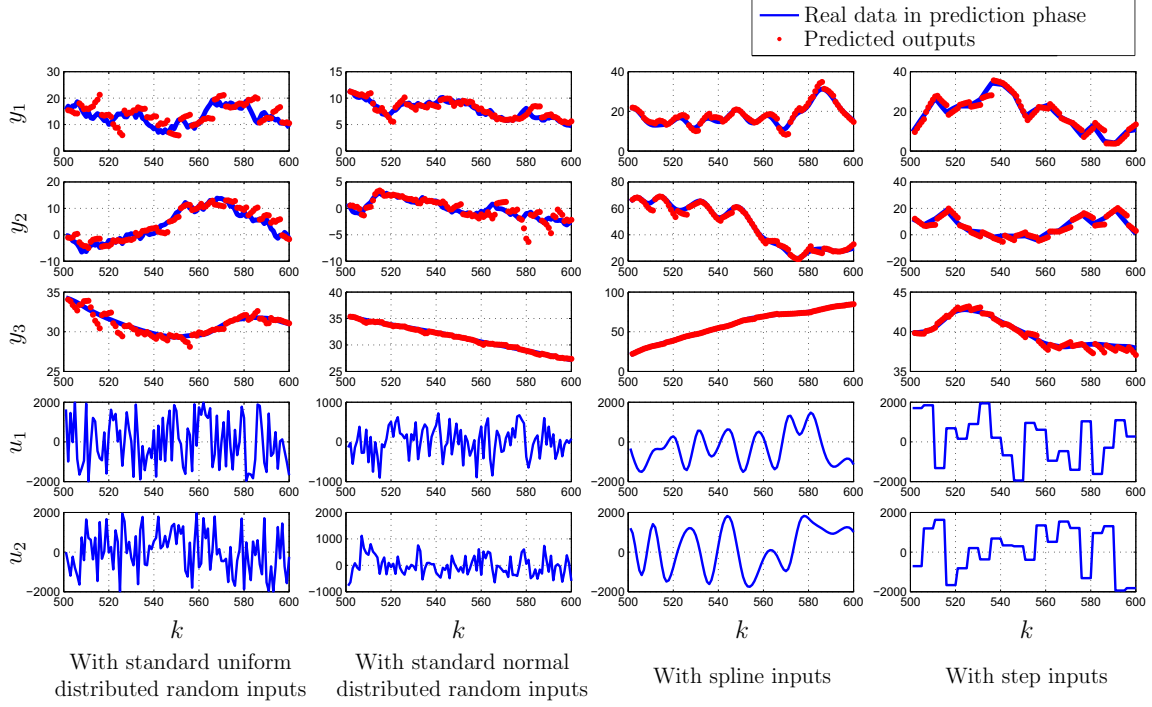


Figure 3.4: Comparison of real outputs and predicted outputs using NARX-RNN with diverse input training data

In order to compare the quality of the prediction results with different test inputs, three normal errors are considered for the evaluation. The first one is the average absolute error (AE). The second one is the mean squared error (MSE) computed with the summarized squared residual by averaging over the real outputs as

$$\text{MSE} = \frac{1}{N_p} \sum_{j=1}^{N_p} (\hat{y}(j) - y(j))^2, \quad (3.7)$$

where $j = 1 \dots N_p$ and N_p is the number of considered prediction steps. The third one is the average absolute relative error (ARE) defined as

$$\text{ARE} = \frac{1}{N_p} \sum_{j=1}^{N_p} \left| \frac{\hat{y}(j) - y(j)}{y(j)} \right|, \quad (3.8)$$

which can be used to compare the prediction accuracy among systems with different scale. The corresponding errors of the prediction phase for each kind of input training data are shown in Table 3.1.

Table 3.1: Prediction errors

Input type	AE	MSE	ARE
Random (uniform)	$\begin{bmatrix} 0.9006 \\ 0.6205 \\ -0.1220 \end{bmatrix}$	$\begin{bmatrix} 11.7063 \\ 4.8331 \\ 0.2717 \end{bmatrix}$	$\begin{bmatrix} 0.0374 \\ 0.5948 \\ 0.0019 \end{bmatrix}$
Random (normal)	$\begin{bmatrix} 0.1299 \\ 0.0273 \\ -0.0477 \end{bmatrix}$	$\begin{bmatrix} 0.5563 \\ 1.3081 \\ 0.0243 \end{bmatrix}$	$\begin{bmatrix} 0.0124 \\ 0.3382 \\ 0.0006 \end{bmatrix}$
Spline	$\begin{bmatrix} 0.5254 \\ -0.6196 \\ -0.0958 \end{bmatrix}$	$\begin{bmatrix} 1.8516 \\ 3.4508 \\ 0.1089 \end{bmatrix}$	$\begin{bmatrix} 0.0104 \\ 0.0048 \\ 0.0006 \end{bmatrix}$
Step	$\begin{bmatrix} 0.0808 \\ -0.1484 \\ -0.0767 \end{bmatrix}$	$\begin{bmatrix} 3.8047 \\ 5.8877 \\ 0.1166 \end{bmatrix}$	$\begin{bmatrix} 0.0193 \\ 0.1213 \\ 0.0011 \end{bmatrix}$

Figure 3.4 and Table 3.1 show a clear prediction quality of NARX-RNN with different kinds of training input data. With all four kinds of input signals, NARX-RNN can predict the system outputs with acceptable accuracy. As a result, the desired new control inputs can be considered for the controller design with an arbitrary kind of these input signals.

- The NARX-RNN can also be used to approximate the inverse relation between the system inputs and outputs [NS14] such as

$$\begin{aligned}
 \hat{u}_d(k+s-1) = & f_2(W_3 \cdot f_1(W_1 \begin{bmatrix} y_d(k+s) \\ y_d(k+s-1) \\ \vdots \\ y_d(k+1) \\ y(k) \\ \vdots \\ y(k+s-l+1) \end{bmatrix} \\
 & + W_2 \begin{bmatrix} \hat{u}_d(k+s-2) \\ \hat{u}_d(k+s-1) \\ \vdots \\ \hat{u}_d(k) \\ \hat{u}(k-1) \\ \vdots \\ \hat{u}(k+s-l-1) \end{bmatrix} + b_1) + b_2),
 \end{aligned} \tag{3.9}$$

where $\hat{u}_d(p)|_{p=k, \dots, k+s-1}$ is the predicted desired system input vector, and

$y_d(p)|_{p=k+1, \dots, k+s}$ the desired system output vector.

3.1.2 Gaussain process regression

Gaussain process regression (GPR) can also be used for identification and multi-step prediction of nonlinear dynamic systems. Different from Neural network, GPR does not learn the system model by adjusting the parameters of a given assumed fix structure, but finds the probability distribution of all possible functions related to the system. The basic idea and the principle of GPR is briefly summarized as outline in [RW06].

Here, a standard linear model with Gaussian noise

$$y_s = f(u) + \epsilon = u^T w + \epsilon, \quad (3.10)$$

where y_s is the output value, $u \in \mathbb{R}^{m \times 1}$ the input vector, $w \in \mathbb{R}^{m \times 1}$ the weights vector of the linear model, and

$$\epsilon \sim \mathcal{N}(0, \sigma_n^2) \quad (3.11)$$

with zero mean and variance σ_n^2 as an additive noise is considered.

The system model can be identified by finding the functions with high probability among all possible random functions describing the system. The probabilities of all possible functions can be computed as the posterior distribution using Bayes' Rule

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}. \quad (3.12)$$

Similar to NARX-RNN, a training phase here is also necessary. During the training phase, the prior distribution is determined by assuming the weights with a Gaussian distribution with zero mean and covariance matrix Σ_p like

$$w \sim \mathcal{N}(0, \Sigma_p). \quad (3.13)$$

The likelihood and marginal likelihood can be computed using the training set including l input-output pairs of the system. Therefore, the posterior can be calculated as

$$p(w|U, y) \sim \mathcal{N}\left(\frac{1}{\sigma_n^2} A^{-1} U y, A^{-1}\right), \quad (3.14)$$

where $U \in \mathbb{R}^{m \times l}$ is the input matrix from the training set, $y \in \mathbb{R}^{l \times 1}$ the output vector from the training set, and $A = \sigma_n^{-2} U U^T + \Sigma_p^{-1}$.

During the prediction phase, the predictive distribution for $f(u_*)$ with a giving new input vector $u_* \in \mathbb{R}^{m \times 1}$ is calculated using

$$p(\hat{f}|u_*, U, y) \sim \mathcal{N}\left(\frac{1}{\sigma_n^2} u_*^T A^{-1} U y, u_*^T A^{-1} u_*\right). \quad (3.15)$$

The corresponding mean value is the predicted system output value.

This is the basic principle of GPR from the weight-space point of view. Unfortunately, this kind of solution has a limited expressiveness. In order to solve this problem, the predictive distribution is rewritten by transforming the input $u \in \mathbb{R}^{m \times 1}$ into a space of powers of u : $\phi(u) \in \mathbb{R}^{l \times 1}$, e.g. for $m = 1$: $\phi(u) = (1, u, u^2, u^3, \dots, u^{l-1})^T$. Correspondingly, the predictive distribution is rewritten as

$$p(\hat{f}|u_*, U, y) \sim \mathcal{N}(\phi_*^T \Sigma_p \Phi (C + \sigma_n^2 I)^{-1} y, \phi_*^T \Sigma_p \phi_* - \phi_*^T \Sigma_p \Phi (C + \sigma_n^2 I)^{-1} \Phi^T \Sigma_p \phi_*), \quad (3.16)$$

where $\phi_* = \phi(u_*)$, $\Phi = \Phi(U)$ as the aggregation of $\phi(u)$, and $C = \Phi^T \Sigma_p \Phi$.

Define the covariance function $c(u, u') = \phi(u)^T \Sigma_p \phi(u')$ with u and u' as arbitrary inputs from the training set or the new input vector, the matrix $C = C(\cdot, \cdot)$ is exact the matrix of the covariances evaluated at all pairs from the corresponding training or prediction set. With the covariance matrix C , both of the training phase (3.14) and the prediction phase (3.16) can be represented more efficiently with a Gaussian process defined as: “a collection of random variables which have a joint multivariate Gaussian distribution [KGBMS05]”.

In detail, (3.15) can be described as

$$f \sim \mathcal{GP}(m(u), c(u, u')), \quad (3.17)$$

where $m(u)$ is the mean function which is usually set to be zero for notational simplicity. Considering the additional noise,

$$y \sim \mathcal{N}(0, C(U, U) + \sigma_n^2 I) \quad (3.18)$$

describes the training process and

$$\begin{bmatrix} y \\ \hat{f} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} C(U, U) + \sigma_n^2 I & C(U, u_*) \\ C(u_*, U) & c(u_*, u_*) \end{bmatrix}\right) \quad (3.19)$$

describes both training and prediction processes.

It is important to emphasize that different kinds of covariance functions exist. Each kind of covariance function has a fix structure with some parameters which are denoted as hyperparameters in the context of GPR. The details about the different covariance functions are discussed later.

In practice, (3.18) can be determined using maximum likelihood method. Defining the log marginal likelihood as

$$\mathcal{L}(\theta) = \log p(y|U) = -\frac{1}{2}y^T(C + \sigma_n^2 I)^{-1}y - \frac{1}{2}\log |C + \sigma_n^2 I| - \frac{l}{2}\log 2\pi, \quad (3.20)$$

the hyperparameters of the covariance function with a given structure are trained by maximizing the $\mathcal{L}(\theta)$ with the training data. The maximization can be realized using any optimization methods as explained in [KGBMS05].

With the calculated hyperparameters, the predictive distribution for the new input u_* can be determined as

$$\hat{f}|U, y, u_* \sim \mathcal{N}(m(\hat{f}), \text{cov}(\hat{f})) \quad (3.21)$$

with

$$m(\hat{f}) \triangleq \mathbb{E} [\hat{f}|U, y, u_*] = C(U_*, U) [C(U, U) + \sigma_n^2 I]^{-1} y \quad (3.22)$$

as the mean and

$$\text{cov}(f) = c(u_*, u_*) - C(u_*, U) [C(U, U) + \sigma_n^2 I]^{-1} C(U, u_*) \quad (3.23)$$

as the variance. The new output is determined as

$$\hat{y}|U, y, u_* \sim \mathcal{N}(m(\hat{f}), \text{cov}(\hat{f}) + \sigma_n^2 I). \quad (3.24)$$

From the forementioned discussion, it can be concluded that the problem of system identification and prediction is actually the problem of finding suitable covariance function with suitable hyperparameters.

Based on the basic idea and principle of GPR, it can also be applied for the identification and multi-step prediction problem for nonlinear dynamic MIMO systems. The related important algorithm is summarized as explained in [GRMS02].

Considering a discrete or a discrete nonlinear dynamic system

$$\begin{aligned} x(k+1) &= [y(k), y(k-1), \dots, y(k-l), \\ &\quad u_*(k+1), u(k), u(k-1), \dots, u(k-l+1)]^T \\ \hat{y}(k+1) &= f(x(k+1)) \end{aligned} \quad (3.25)$$

with a smooth assumed function f , the s -step ahead prediction is achieved by repeating the one-step ahead prediction for s times iteratively.

For $i = 1$, the predicted output $\hat{y}(k + 1)$ is calculated using

$$\begin{aligned} x(k + 1) &\sim \mathcal{N} \left(\begin{bmatrix} m(x(k)) \\ \vdots \\ m(x(k - l + 1)) \end{bmatrix}, \right. \\ &\quad \left. \begin{bmatrix} v(x(k)) & \cdots & \text{cov}(y(k), u(k - l + 1)) \\ \vdots & \ddots & \vdots \\ \text{cov}(u(k - l + 1), y(k)) & \cdots & v(x(k)) \end{bmatrix} \right) \\ \hat{y}(k + 1) &\sim \mathcal{N}(m(x(k + 1)), v(x(k + 1))) \end{aligned} \quad (3.26)$$

and for $i = 2, 3, \dots, s$, the predicted output at $k + i$ is calculated using

$$\begin{aligned} \hat{x}(k + i) &\sim \mathcal{N} \left(\begin{bmatrix} m(\hat{x}(k + i - 1)) \\ \vdots \\ m(x(k + i - l)) \end{bmatrix}, \right. \\ &\quad \left. \begin{bmatrix} v(\hat{x}(k + i - 1)) & \cdots & \text{cov}(\hat{y}(k + i - 1), u(k + 1 - l)) \\ \vdots & \ddots & \vdots \\ \text{cov}(u(k + 1 - l), \hat{y}(k + i - 1)) & \cdots & v(x(k + i - l)) \end{bmatrix} \right) \\ \hat{y}(k + i) &\sim \mathcal{N}(m(\hat{x}(k + i)), v(\hat{x}(k + i))). \end{aligned} \quad (3.27)$$

Considering $x_* \in x(k + i - j)$ with $j = 1, \dots, l$, $m(x_*)$ and $v(x_*)$ are computed as

$$m(x_*) = E_{x_*}[\mu(x_*)] \quad (3.28)$$

$$v(x_*) = E_{x_*}[\sigma^2(x_*)] + \text{var}_{x_*}(\mu(x_*)) \quad (3.29)$$

with

$$\begin{aligned} \mu(x_*) &= C(x_*, x)^T C(x, x)^{-1} [y(k - l + 1), y(k - l + 2), \dots, y(k)]^T \\ \sigma^2(x_*) &= c(x_*, x_*) - C(x_*, x)^T C(x, x)^{-1} C(x, x_*). \end{aligned} \quad (3.30)$$

As mentioned before, with a given structure of covariance function, the concrete covariance function $c(x_*, x_*)$ as well as the covariance matrix $C(x, x)$ are determined by maximizing the log marginal likelihood (3.19).

Several points should be additionally mentioned regarding the use of GPR for prediction of nonlinear dynamic MIMO systems. In order to clearly state these points, simulation examples for application of GPR to the Lorenz-system (3.5) are discussed in the following.

- Gaussian process regression can only be directly used for MISO systems. In order to apply it for MIMO systems, the prediction phase can be repeated to predict each output of the system which is treated as independent from the other outputs [RW06].
- Using GPR, the suitable value of the prediction horizon size s should also be determined experimentally for different systems according to the acceptability of the prediction accuracy. Assuming the covariance function is taken as

$$c = C(u, u') = v_1 \exp \left[-\frac{1}{2} \sum_{d=1}^m w_d (u^p - u^q)^2 \right] + v_0 \quad (3.31)$$

with the hyperparameter $w_1 \dots w_D$, v_0 , and v_1 as well as $s = 50$, $T_S = 10^{-3}s$, the standard deviation of the noise $s_n = 0.05$, $T_t = 0.5s$, and the initial condition of the outputs $x_0 = [-10 \ 10 \ 25]$, the training process and prediction result by using GPR for the Lorenz-system (3.5) with standard normal distributed random training and test input signals is shown in Figure 3.5.

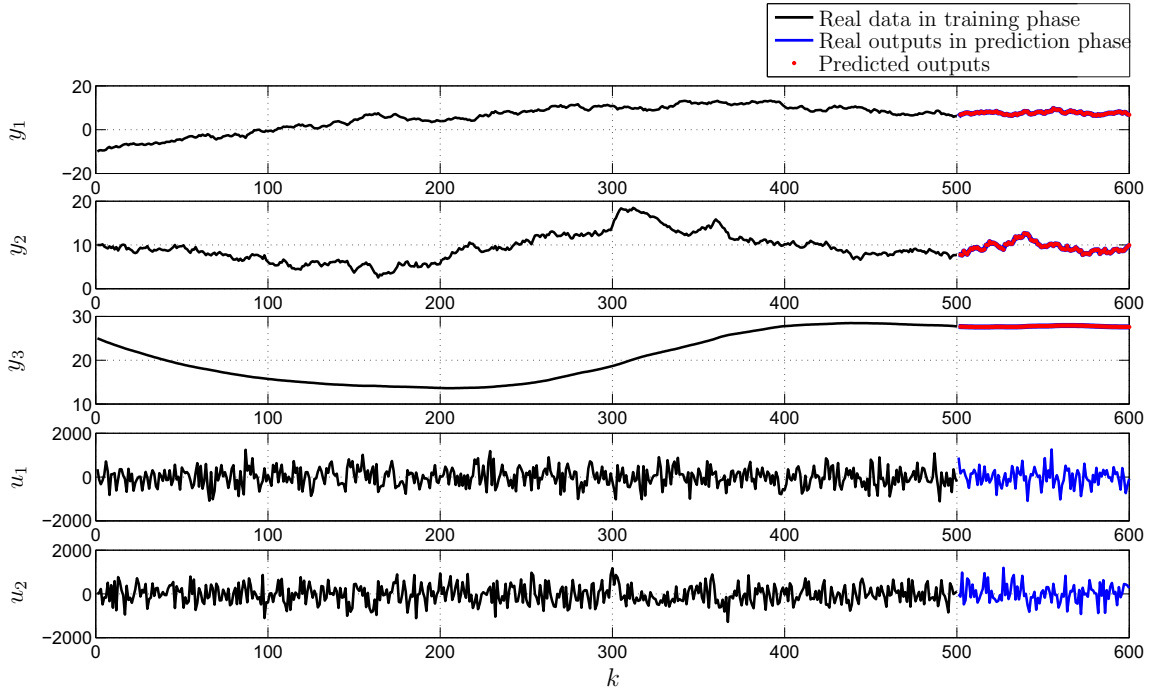


Figure 3.5: Comparison of real outputs and predicted outputs using GPR with standard normal distributed random input training and test data

The two input and three output data in the training phase (the first 500 steps) are denoted by black lines. The blue lines show the test data of the inputs and outputs in the prediction phase. The predicted outputs denoted by the

red points clearly indicate that GPR with standard normal distributed random input training data can be used for predicting the outputs of nonlinear dynamic MIMO systems with high accuracy. Furthermore, in this example, there is no need to repeat the identification and prediction process as in the case with NARX-RNN for the 0.1 second prediction process, because the suitable value of s in this case is about 100. This indicates that the system outputs can be predicted using GPR for much more steps in the future than using NARX-RNN.

- During the training phase, the input data can also be generated in different way. For example as random signals with standard normal distribution [KGBMS05], as standard uniform distribution [WR96], or as step functions [SWMST07]. In order to decide the type of input data in practice, it is necessary to check the performance of the GPR with different typical training input data.

Taking the same assumptions as last case, the corresponding simulation results of the prediction phase by applying GPR for the Lorenz-system (3.5) with different kinds of input training data are shown together in Figure 3.6.

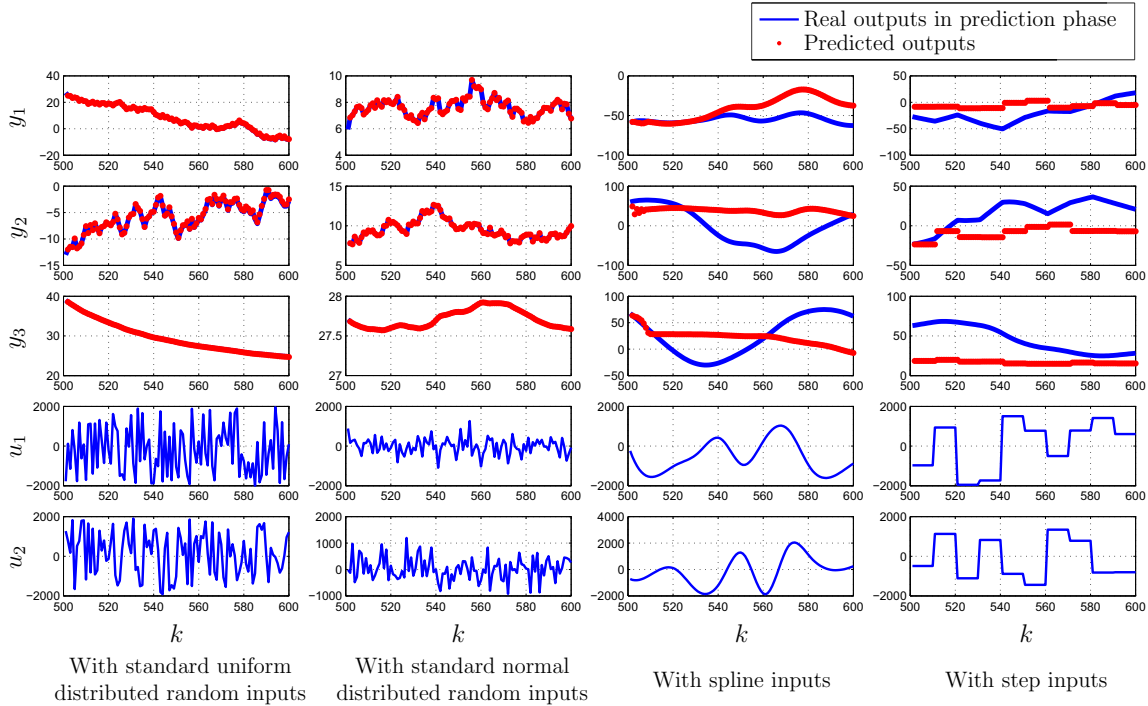


Figure 3.6: Comparison of real outputs and predicted outputs using GPR with diverse input training data

The corresponding error analysis is shown in Table 3.2.

Table 3.2: Prediction errors

Input type	AE	MSE	ARE
Random (normal)	$\begin{bmatrix} -0.0970 \\ 0.1504 \\ -0.0599 \end{bmatrix} * 10^{-3}$	$\begin{bmatrix} 0.0988 \\ 0.3097 \\ 0.0544 \end{bmatrix} * 10^{-7}$	$\begin{bmatrix} 0.2096 \\ 0.2672 \\ 0.0364 \end{bmatrix} * 10^{-5}$
Random (uniform)	$\begin{bmatrix} 0.2154 \\ 0.1389 \\ -0.0233 \end{bmatrix}$	$\begin{bmatrix} 0.0843 \\ 0.0318 \\ 0.0017 \end{bmatrix}$	$\begin{bmatrix} 0.0226 \\ 0.0070 \\ 0.0002 \end{bmatrix}$
Step	$\begin{bmatrix} 15.0089 \\ -25.1663 \\ -28.6051 \end{bmatrix}$	$\begin{bmatrix} 0.5332 \\ 0.8069 \\ 1.0596 \end{bmatrix} * 10^3$	$\begin{bmatrix} 0.1612 \\ 0.2455 \\ 0.0967 \end{bmatrix}$
Spline	$\begin{bmatrix} 13.4188 \\ 40.3143 \\ -2.6226 \end{bmatrix}$	$\begin{bmatrix} 0.3306 \\ 3.3769 \\ 2.0425 \end{bmatrix} * 10^3$	$\begin{bmatrix} 0.0436 \\ 0.5105 \\ 0.5282 \end{bmatrix}$

Figure 3.6 and Table 3.2 show a clear prediction quality of GPR with different training input data. With standard normal distributed random training input data, GPR can predict the system outputs with high accuracy. With standard uniform distributed random training input data, the prediction result is acceptable but the accuracy is lower than the normal distributed random inputs. This kind of performance of GPR is logic, because the system is described as a Gaussian process, which can describe the data with Gaussian/normal distribution better than with non-Gaussian distribution. With step functions as the training input data, the prediction results are not acceptable. With spline functions, the system outputs can only be predicted for few steps and the accuracy of the prediction can not be guaranteed.

It is necessary to be denoted, the prediction results are not always acceptable for every output although random (uniform or normal distributed) training data are used. Taking the same assumption as above and repeating the simulation for several times, the prediction result can be unacceptable for one output as shown in Figure 3.7.

The reason of such phenomena can be explained from the practical point of view. Using GPR, the system identifier needs enough important training data from the system, which can not be guaranteed by a random input training data. If all training data present the system dynamics under the same or similar condition, GPR can logically not learn the dynamic of the system under a new condition. The performance of GPR with respect to the training data can accordingly be concluded. Denote the knowledge of the system dynamics under different conditions as the information about the system, GPR can only

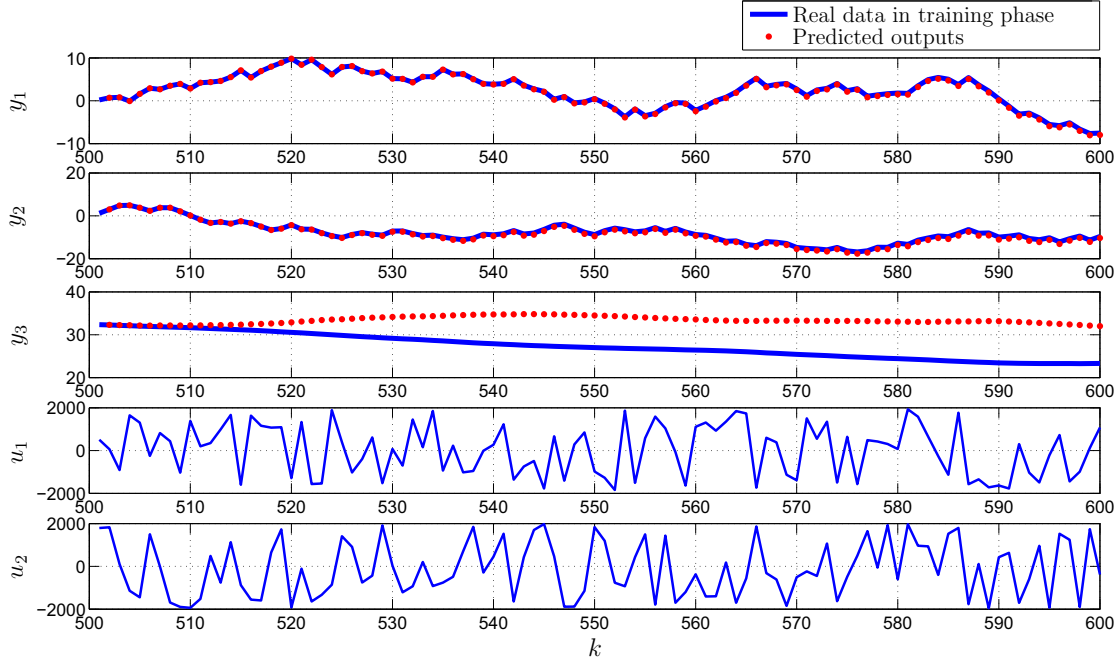


Figure 3.7: An unacceptable prediction using GPR with standard uniform distributed random input training data

predict the system outputs with high accuracy with the training data including enough information about the system. The random training data randomly and the step or spline input training data usually do not include enough system information. Because the unknown nonlinear systems are considered in this thesis, suitable training data which contain enough information about the system can not be predefined. As a result, the standard normal distributed random training data are applied for the prediction task and a validation process is necessary to verify the acceptability of the prediction results online.

- The negative log marginal likelihood [RW06] defined as

$$LL = \frac{1}{2} \log |C| + \frac{1}{2} y^T C^{-1} y + \frac{l}{2} \log(2\pi) \quad (3.32)$$

can be used to validate the quality of the training process. The smaller the LL is, the better the training process is. If the LL value of the training process is not small enough, which means $LL < \delta^*$ denoted as a threshold of LL that can be determined after some training tests for a certain system, it is necessary to take other input data for the training process.

- As mentioned before, the structure of the covariance function of GPR should be predefined. Some possible covariance functions are given in Table 3.3.

Table 3.3: Possible covariance functions

Type	Equation ($c(x, x') =$)	Hyperparameters
Constant	σ_f^2	$\ln(\sigma_f)$
Linear	$x^T x'$	\emptyset
Linear with diagonal weightings	$x^T \Lambda^{-2} x'$	$\ln(\lambda_1), \dots, \ln(\lambda_D)$
Squared exponential	$e^{-\frac{1}{2l^2} x^T x'}$	$\ln(l)$
Diagonal squared exponential	$e^{-\frac{1}{2l^2} (x-x')^T (x-x')}$	$\ln(l), \ln(\sigma_f)$
Full squared exponential	$\sigma_f^2 e^{-\frac{1}{2} (x-x')^T \Lambda^{-2} (x-x')}$	$\ln(\lambda_1), \dots, \ln(\lambda_D), \ln(\sigma_f)$
Rational quadratic 1	$\sigma_f^2 (1 + \frac{1}{2\alpha} (x-x')^T \Lambda^{-2} (x-x'))^{-\alpha}$	$\ln(\lambda_1), \dots, \ln(\lambda_D), \ln(\sigma_f), \ln(\alpha)$
Rational quadratic 2	$\sigma_f^2 (1 + \frac{1}{2\alpha l^2} (x-x')^T (x-x'))^{-\alpha}$	$\ln(\sigma_f), \ln(\alpha), \ln(l)$

The covariance functions in Table 3.3 can be used individually or as combination with each other. Some simulation results are obtained to check the performance of using GPR for the Lorenz-system with several different covariance functions. Assuming $s = 50$, $T_S = 10^{-3}s$, $s_n = 0.05$, $T_t = 0.5s$ and the training input data and test input data are for each simulation the same standard normal distributed random data within the interval $[-2000 \ 2000]$, the simulation results of the prediction phase are shown together in Figure 3.8.

The prediction results are different although the simulations have the same condition except the covariance function. Only the rational quadratic covariance function used for the GPR gives an acceptable prediction result for the Lorenz-system. Therefore, it is necessary to experimentally find a suitable covariance function for a certain system before applying the GPR in the prediction process.

- Due to the fact that GPR can only be applied directly to MISO systems, it can not be used to identify or predict the inverse MIMO systems.

3.1.3 Combined identifier

As pointed out, none of the existing identification method is suitable for all kinds of systems. There are always some constrains and shortcomings during the application. For example, the PBSID method is not suitable for identifying non-Hammerstein-Wiener nonlinear MIMO systems. Therefore the predicting result using PBSID for Lorenz-system is not acceptable as shown in Figure 3.9.

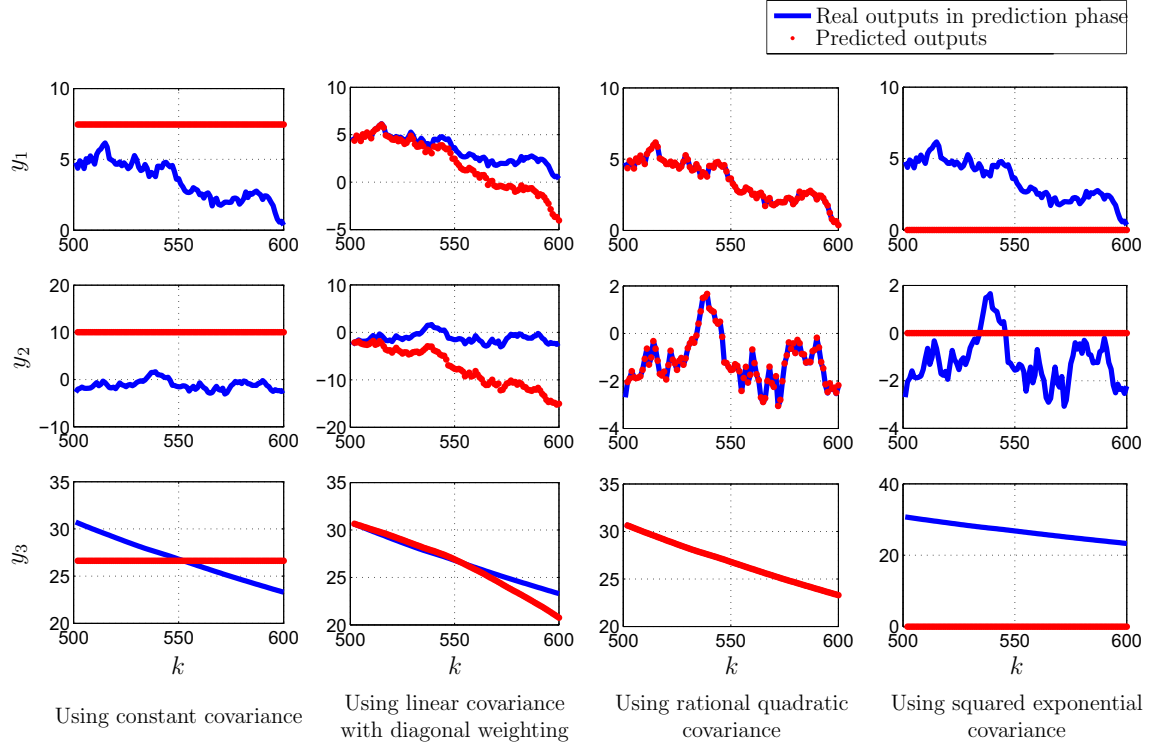


Figure 3.8: Comparison of real outputs and predicted outputs using GPR with diverse covariance functions

Another example is shown in the subsections 3.1.1 and 3.1.2 that both NARX-RNN and GPR identification methods have their own advantages and disadvantages. In particular, NARX-RNN can predict the system outputs with acceptable accuracy for each kind of system input data and can be used directly for MIMO systems and its inverse model: GPR can predict the system outputs with higher accuracy but only with random input data and can not be used directly for MIMO systems. If the more suitable method can be selected automatically for predicting the system dynamic behavior under different situations, the high autonomous can be achieved for the module “perception and interpretation” of the cognitive framework. In order to achieve the identification task for most kinds of systems with good performance, a combined identifier is designed based on the existing identification methods.

As mentioned before, a tuning algorithm in the combined identifier should be designed suitable for the online process. Taking NARX-RNN and GPR as the basis identifiers, the principle idea of combined identifier developed in this thesis is explained using a flowchart as shown in Figure 3.10.

During the initial training process, GPR is first used to identify the system dynamic behavior with standard normal distributed input training data and the corresponding system output training data are measured from the unknown system, because

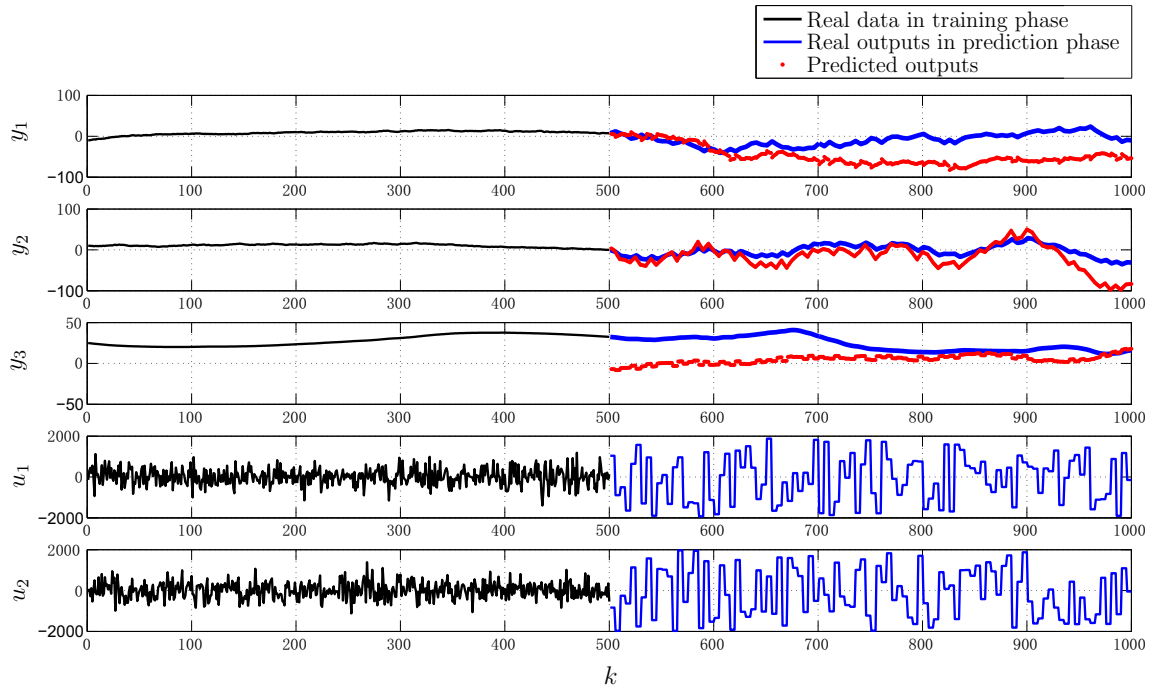


Figure 3.9: Comparison of real and predicted outputs using PBSID

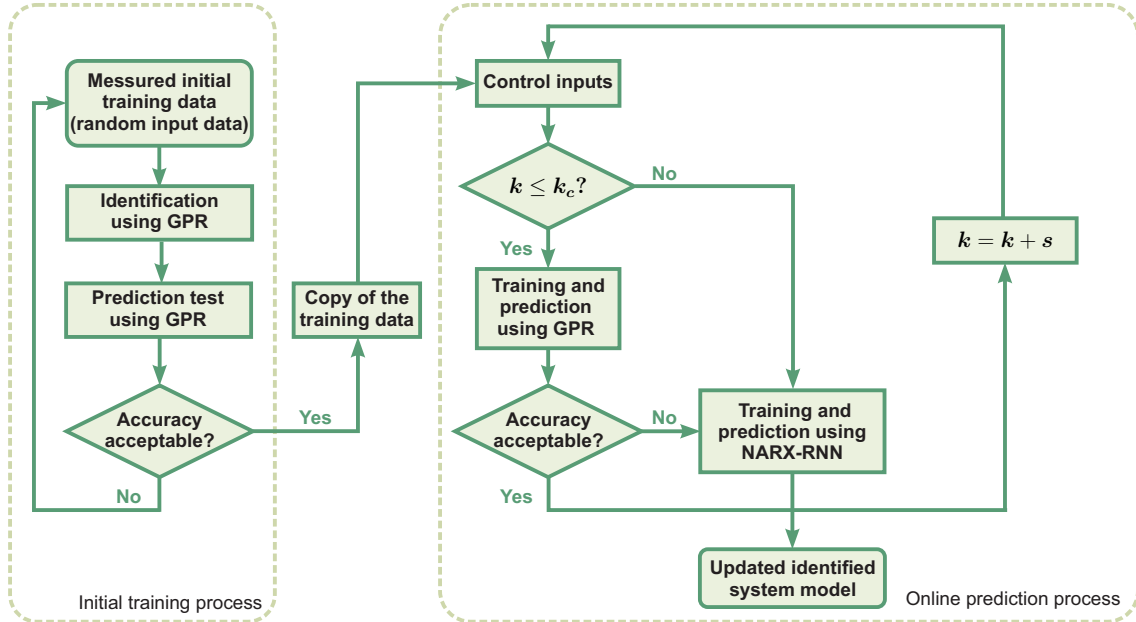


Figure 3.10: Flowchart of the combined identifier

GPR can normally identify the unknown system dynamics with higher accuracy if the input training data can be taken arbitrarily. If the negative log marginal likelihood of the identified system outputs is not small enough, the covariance function of the GP and its corresponding hyperparameters should be changed until the log marginal likelihood is small enough.

After the initial training (the identification) process, the identified Gaussian process with its covariance function and hyperparameters are used to predict the system dynamics with certain test input data and the corresponding test output data. Considering the whole cognitive stabilizer, the type of test input data should be defined according to the form of the control inputs generated in the modules “planning and execution”. For example, the control inputs are determined according to the control goal for each step separately, which means that the value of the control input vectors are not always the same for each step and the test input data should be a standard uniform distributed random signals in order to have similar form as the control inputs. Similarly, if the control inputs with the same value are determined for each s steps, the test input data should be step function signals.

The test prediction result is evaluated according to the ARE. If the ARE of the predicted system outputs is not small enough, the training input and output data may not have enough information about the system dynamic behavior. In this case, the whole training process should be repeated until a suitable training data are found. Otherwise, the training data should be saved and used for the online prediction (the test input and output data are not more needed).

Applied for the online prediction process, GPR is used to predict the system outputs with the control inputs for the first k_c steps, which can be predefined or set experimentally (usually set as $0.2 \cdot T_t \cdot T_s$). Similar to using NARX-RNN or GPR, the training phase and prediction phase are repeated for each s steps in order to achieve online identification and prediction of the system dynamic behavior. The system inputs and outputs in the last $T_t \cdot T_s$ steps are considered as the training data to predict the system outputs for the next s steps. For the first k_c steps, the most part of the system inputs are standard normal distributed random inputs. Therefore the GPR is taken as the training and prediction method due to its better performance of prediction accuracy than NARX-RNN in such case. However, in order to guarantee the prediction accuracy, an additional validation process is used to verify whether the GPR can predict the system outputs with acceptable accuracy. Due to the fact that the real system outputs for the future steps can not be measured in the online prediction process, the equation (3.6) based only on the predicted system outputs and the real system outputs in the past is used as the condition for unsuitable prediction results. If the prediction results are not acceptable, NARX-RNN is used for the same prediction task. If the new prediction result using NARX-RNN is better (evaluated using equation (3.6)) than the old one using GPR, the new one is treated as the suitable one. Otherwise the old one is taken as the predicted result.

After k_c steps, only NARX-RNN is considered for the online prediction process if the prediction result using GPR should always be repeated using NARX-RNN.

Using this strategy, the suitable identification method can be selected automatically in the online process with guaranteed accuracy. Taking the assumption of the sample time $T_s = 10^{-3}s$, the training time $T_t = 0.5s$, $\eta = 30$, $\alpha = 0$, $s_n = 0.05$, the initial condition of the outputs $x_0 = [-10 \ 10 \ 25]$, $s = 5$, and the covariance function (3.31), the cognitive identifier is applied to the Lorenz-system (3.5) with a spline input training data and step control input data. The corresponding training phase and the prediction results are shown in Figure 3.11 for evaluating the performance of the combined identifier.

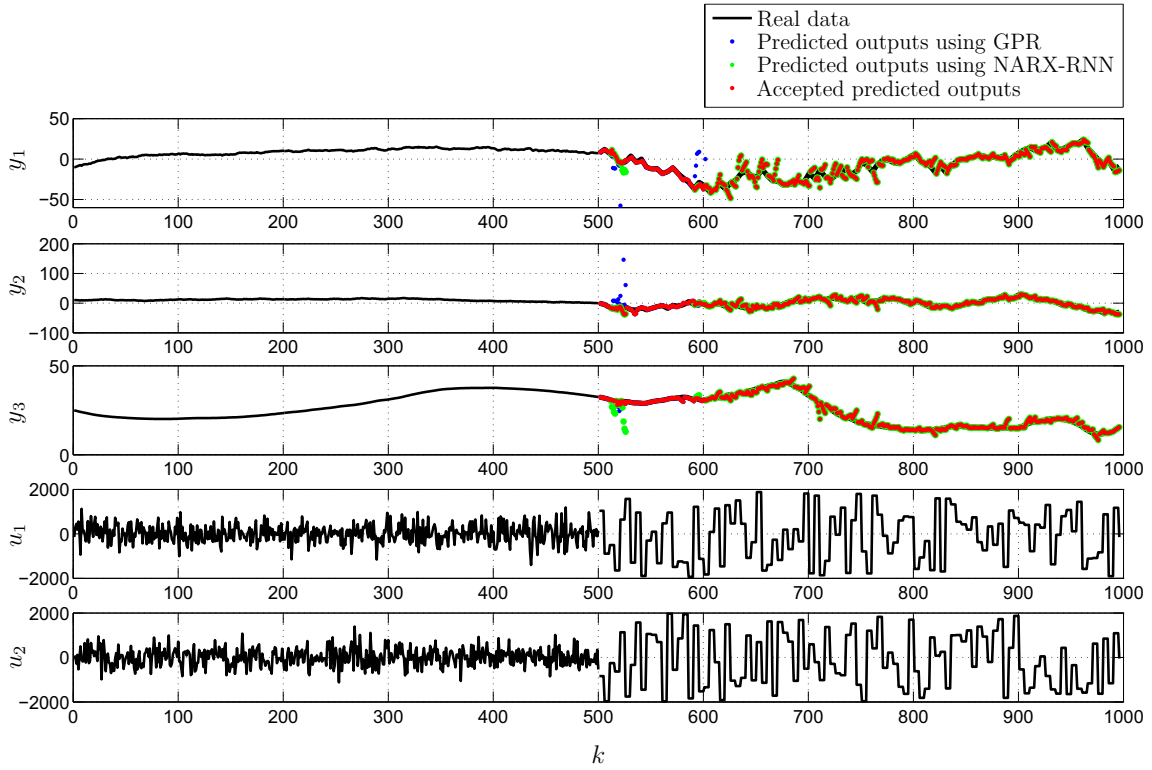


Figure 3.11: Comparison of real and predicted outputs using combined identifier

The input and real output data are denoted with black lines. The predicted system outputs are shown with blue points when using GPR and with green points when using NARX-RNN. The accepted predicted outputs denoted with the red points indicate clearly that the combined identifier can be used for predicting the outputs of nonlinear dynamic MIMO-system with acceptable accuracy.

For $k = 500$ to about $k = 600$, the system outputs are predicted using GPR with high accuracy except at $k = 515$, $k = 520$, $k = 525$, and $k = 590$, as denoted

with the blue points. Therefore NARX-RNN is used for such cases to guarantee the prediction accuracy. Only the unacceptable predicted outputs using GPR are replaced with the predicted result using NARX-RNN. For example for $k = 515$, only the first and second predicted outputs using GPR are not suitable enough, therefore they are replaced by the predicted result using NARX-RNN and there is no need to take the predicted result using NARX-RNN for the third output. After $k = 600$, only NARX-RNN is used and the corresponding results are acceptable. This strategy is also shown with the absolute prediction error in Figure 3.12.

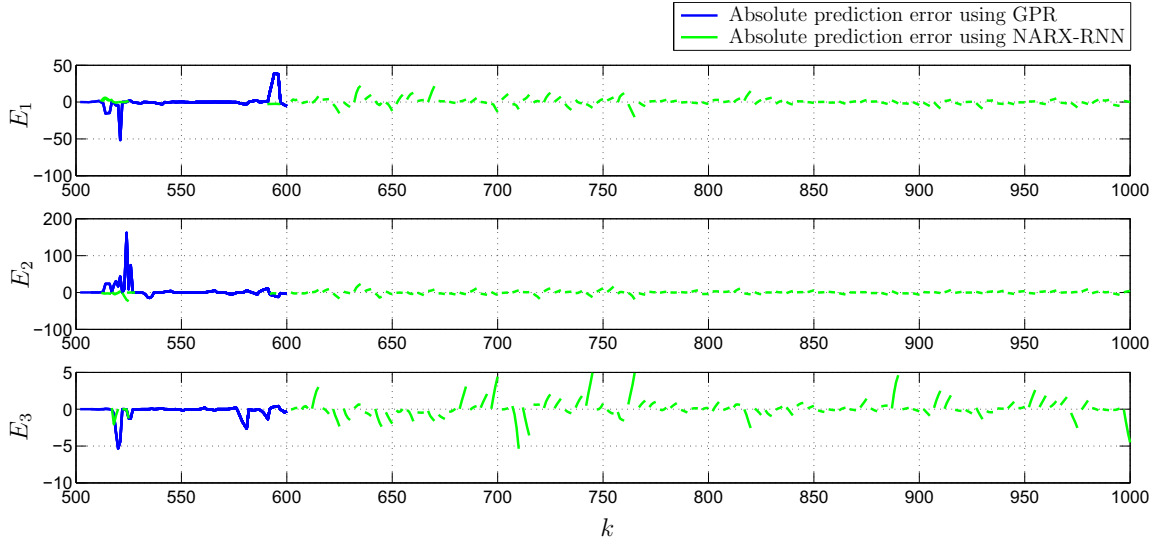


Figure 3.12: Comparison of the absolute prediction error using GPR and NARX-RNN

The red lines denote the prediction error using GPR and the blue lines denote the prediction error using NARX-RNN. It can be detected from the figures, that for the first 100 prediction steps, the unacceptable predicted system outputs using GPR can be detected and corrected using NARX-RNN during the online prediction process.

It can be concluded that using this kind of combined identifier, the prediction accuracy can be guaranteed (using validation processes) and optimized (using GPR for the possible cases).

3.2 Realization of module “expert knowledge”

As mentioned in chapter 2, suitable stability criteria stored in the module “expert knowledge” should be able to be used in the data-driven manner in real time. Furthermore, the stability criteria should be suitable for switched systems defined as

$$x(k+1) = f_i(x(k)), \quad i \in [1, \dots, N], \quad (3.33)$$

where N is the number of the different descriptions of the system. The reason for this requirement is that the system to be controlled is assumed as unknown system and is identified using online system identification for each s steps. In other words, there is a new description of the system for each s steps and the system switches between the old description and the new description for each s steps. Although the system can be proofed as stable system for each f_i , the stability of the whole switched system can not be always guaranteed. A typical example is given in [Bra98]. As a result, stability criteria which are suitable for switched systems and used in the data-driven manner in real time are needed for the cognitive framework.

Up to now, very few stability criteria with such capability have been developed. In this thesis, data-driven quadratic stability criterion [ZS14], quadratic Lyapunov stability criterion with a certain Lyapunov function [NS14], and uniform stability of switched systems are considered and separately explained in detail in this section.

For the first two stability criteria, the following definitions and theorem proposed in [Bra98] are applied to guarantee the stability of the switched discrete systems with the help of the Lyapunov functions.

It is noted that v is a candidate Lyapunov function which is a positive definite function about the origin ($v(0) = 0$), K is a strictly increasing sequence of time steps $K = [k_1, k_2, \dots, k_N, \dots]$ where $k_j \in K$ with $j \in \mathbb{Z}^+$ indicates the switching time step between two descriptions of the system model, $\mathcal{I}(i)$ is the set of the time steps that the i th system description is active and $x_i(\cdot)$ is the corresponding all possible trajectories (with respect to all possible initial states), as well as $K(i) = K \cap [k_1 - 1, k_2 - 1, \dots, k_N - 1, \dots]$ and $\chi(K)$ indicate the set of $\mathcal{I}(i) \cap K$.

Definition 3.1: v_i is Lyapunov-like for function f_i and trajectory $x_i(\cdot)$ over K_i if:

- $v_i(x(k+1)) \leq v_i(x(k))$ for all $k \in \mathcal{I}(i)$;
- v_i is monotonically nonincreasing on $\chi(K)$.

Theorem 3.2: Suppose candidate Lyapunov functions v_i , $i = 1, \dots, N$ and switched system (3.33) with $f_i(0) = 0$ for all i exist. If for each possible switching sequence with all possible initial state, v_i is Lyapunov-like for f_i and $x_i(\cdot)$ over K_i , then the system is stable in the sense of Lyapunov.

Here, it is assumed that $x(k) \in \mathbb{R}^n$ and each f_i is globally Lipschitz continuous. With the measured or predicted state $x(k)$, the condition $x(k) \in \mathbb{R}^n$ can always be fulfilled in the context of cognitive framework. With the help of the validation process (3.6) in the identification process, the globally Lipschitz continuity of each f_i can also be guaranteed, because Δx is globally bounded.

In the context of cognitive stabilizer, f_i is identified online for each s steps, which means that the i th description is only active for these s steps. Therefore, K can be

denoted here as $K = [k_1, k_2, \dots, k_N]$. As a result, there is always only one element in the set $\chi(K)$ and the *Definition 3.1* can be simplified here as

Definition 3.3: v_i is Lyapunov-like for function f_i and trajectory $x_i(\cdot)$ over K_i if $v_i(x(k+1)) \leq v_i(x(k))$ for all $k \in \mathcal{I}(i)$.

3.2.1 Data-driven quadratic stability criterion

Data-driven quadratic stability criterion is one of the suitable stability criteria for the cognitive context. This kind of criterion requires only the system states as its inputs to define the stability of the system to be controlled as its output, which means it establishes a direct relationship between the measured system states and the system stability. The criterion detailed in the sequel has been previously been published in [ZS14] and is therefore briefly summarized here.

Data-driven quadratic stability criterion judges the quadratic stability of trajectories of nonlinear discrete-time MIMO systems by utilizing a geometric method. It is assumed, that the system states are fully measurable and free of noise, meaning $y = x$.

Quadratic stability has been defined [Bar85] as: “The system described by $x(k+1) = f(x(k))$ is called quadratic stable if there exists a positive definite matrix P , such that along the solution of the nonlinear discrete time system the function

$$v(x(k)) = x(k)^T P x(k) \quad (3.34)$$

satisfies

$$\Delta v(x(k)) = v(x(k+1)) - v(x(k)) \leq 0.” \quad (3.35)$$

In order to clearly explain the geometric method used by the data-driven quadratic stability criterion, some geometric definitions [ZCZL11] utilized here are necessary to be stated at first.

Definition 3.4: “A negative open half space h^- [Mei93] is the half space of the n -dimensional Euclidean space E^n containing the complete set of points below a non-vertical hyperplane h in E^n .

If a vector w is located in a negative open half space, its inner products with all the vectors located in the space R^{n+} are less than zero.

Definition 3.5: A subset C of a vector space V is a *convex cone* if and only if $p_1x + p_2y$ belongs to C , for any positive scalars p_1, p_2 , and any x, y in C .”

With these geometric definitions, the quadratic stability of the system states can be judged using a geometric approach as given in the sequel.

The controlled system

$$x(k+1) = f(k, x, u(k, x)) = f'(k, x) \quad (3.36)$$

and an arbitrary physical possible state x_e of the system to be controlled are considered. The system to be controlled should be stabilized at the state x_e which is also an equilibrium point of the controlled system. According to the first definition, only the stability of the original point $x_0 = 0$ of the space h can be evaluated. Therefore the state variables are firstly transformed into a new coordinate as $x_{new} = x_{old} - x_e$. In this case, $x_{new} = x_0 = 0$ defines exactly $x_{old} = x_e$.

The data-driven quadratic stability criterion is now expressed with the definitions mentioned above and transformation as: The point $x_0 = 0$ is an asymptotically stable equilibrium point of the system related to the motion observed, if and only if every new vector w is contained in a convex cone which is located in a negative open half space h^- . A similar proof regarding the controlled system is given in [ZS14]. The new vector w is obtained by the transformation of all system states x , $x \neq 0$ with

$$w = \text{diag}[\varphi f(x)]\varphi x \quad (3.37)$$

and φ as an orthogonal matrix.

For example, a 2-dimensional system can be mapped to the new coordinates w_1, w_2 as shown in Figure 3.13. The transformed system states are denoted as the crosses and build a smallest convex conic cone C . The system is quadratic stable, if and only if C is located in a negative open halfspace h^- .

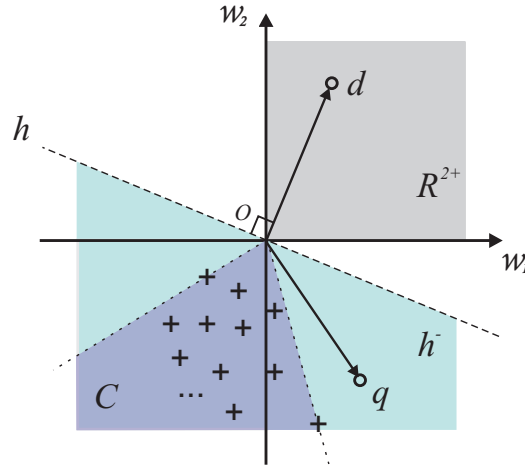


Figure 3.13: Convex conic cone and negative open halfspace [ZS14]

It is evident, that this criterion sets up a relationship between the system states and the stability of the motion of the system, which means that this criterion can be used

in a data-driven manner. As mentioned before, the model of the unknown system is identified using the last l steps and the system states can be predicted for the next s steps. This means that the stability check is based on the trajectory denoted as J_t according to the measured system states for last l steps and the predicted system states for the next s steps. The Data-driven quadratic stability criterion provides in the context of cognitive stabilizer a method to design a suitable controller which can guarantee the stability of the closed-loop system over every possible J_t . Using such a controller, there is no dependency on the trajectory and the initial state. As a result, a quadratic Lyapunov function can be found for the current description of the system and all possible trajectories. So the conditions for the stable switched system in the *Theorem 3.2* can be fulfilled. Therefore this criterion is suitable for the cognitive stabilizer.

3.2.2 Quadratic Lyapunov stability criterion with a certain Lyapunov function

Another possibility to realize the quadratic stability in a data-driven manner is to design a Lyapunov function which can satisfy the requirement given in equations (3.34) and (3.35) only using the system measurements.

The function $v(x(k))$ here is actually the Lyapunov function for discrete time system. Evidently, with a positive definite matrix P ,

$$\begin{aligned} v(x(k)) &= x(k)^T P x(k) > 0, \text{ for } x(k) \neq x_0, \\ &= 0, \text{ for } x(k) = x_0 \end{aligned} \quad (3.38)$$

can be considered as the requirement for establishing the Lyapunov function.

A suitable Lyapunov function as illustrated in [NS14] is

$$v(e(k)) = e^T(k) \cdot e(k), \quad (3.39)$$

using $e(k) = w_d - x(k)$ with w_d as the reference vector of the system outputs. A transformation of the state variables $x_{new} = x_{old} - x_e$ into the new coordinates similar to the data-driven quadratic stability criterion is also necessary. With this transformation, the reference vector can always be defined as $e(t) = -x(t)$. The Lyapunov function is redefined in the new coordinate as

$$\begin{aligned} v(e(k)) &= x^T(k) \cdot x(k), \\ &= \|x(k)\| > 0, \text{ for } x(k) \neq x_0, \\ &= 0, \text{ for } x(k) = x_0, \end{aligned} \quad (3.40)$$

which satisfies the conditions in equation (3.35) with all system states.

The resulting problem is to find the suitable vector $x(k+1)$ to satisfy the condition (3.36) as

$$\|x(k+1)\| \stackrel{!}{<} \|x(k)\|. \quad (3.41)$$

Similar to the data-driven quadratic stability criterion, this criterion can also provide a method to design a suitable controller in the context of cognitive stabilizer. From the control point of view, a suitable range of the desired system states $x_d(k+1)$ can be determined using the inequality (3.41), which can guarantee the quadratic stability of the closed-loop system over J_t . This method has also no dependency on the kind or value of J_t and therefore satisfies the condition in the *Theorem 3.2*. The new control input can be found with the help of the inverse control technique according to the suitable range of $x_d(k+1)$. This kind of realization can only be used together with the inverse learning process as given in section 3.1.

3.2.3 Uniform stability criterion for switched nonlinear systems

The uniform stability of switched nonlinear systems is defined and proofed in [LLX09] and can be considered as an existence of a common Lyapunov function. The definition of uniformly stable is given as

Definition 3.6: “The switched system (3.33) is said to be uniformly stable on N if, given any $\epsilon > 0$, there exists some $\rho = \rho(\epsilon) > 0$, independent of k_0 and $i \in N$, such that $\|x(k_0)\| < \rho$ implies $\|x(k)\| < \epsilon$, for (x, i) with $i \in N$ that solves (3.33).” [LLX09]

There are some theorems given in [LLX09] for the application of this kind of criterion with the help of suitable Lyapunov functions. Another possibility of the application is developed for the cognitive stabilizer and explained in the following.

- It is assumed that the system to be controlled has a physical bounded range R_a of the system states. In other words, the possible initial states always satisfy $\|x(k_0)\| \leq \|R_a\|$.
- The desired equilibrium point of the closed-loop system x_e is transformed as the original point of a new coordinate. The transformed system states $x_{new}(k) = x(k) - x_e$ is desired to be located at the original point.
- For an arbitrary given $\epsilon > 0$, it can be set, that $\rho = \epsilon$. If a suitable controller can be found which can make the transformed predicted system states $\|x_{new}(k)\|$ always located in the given range ϵ , the uniform stability of the closed-loop system can be guaranteed according to *Definition 3.6*.

- During the online control process, the value of $\|x_{new}(k_0)\|$ is always be given from the real system. In order to satisfy the condition $\|x_{new}(k)\| < \epsilon$ for all k in this case, which means the $\|x_{new}(k_0)\|$ should also be smaller than ϵ , an arbitrary value of ϵ should be taken from the set $\epsilon \in [\|x_{new}(k_0)\|, \|R_a - x_e\|]$ with respect to the boundary of the system states. Therefore, if ρ is set to be equal to $\|x_{new}(k_0)\|$, and a suitable controller can be found which can make the transformed predicted system states $\|x_{new}(k)\|$ always located in the given range $\rho \leq \epsilon$, the uniform stability of the closed-loop system can be guaranteed according to *Definition 3.6*.

It should be noted that there is no dependency of the identified system model in this approach and only the system states are needed.

All the three proposed stability criteria are suitable for the data-driven stability evaluation for switched nonlinear system. The data-driven quadratic stability criterion is used to evaluate arbitrary system states at each time step but with a high computational requirement. Using second and third criteria, a suitable exactitude range of the desired system outputs can be calculated. This is useful for choosing the corresponding upcoming control input values efficiently from stability point of view. They can be applied with less computational requirement. However, the inverse model of the system has to be generated first using the second criterion. The detailed application of these stability criteria in the cognitive stabilizer is explained in the Section 3.4.

3.3 Realization of module “planning and execution”

The input values of a system are bounded from the physical point of view in a certain interval denoted by $[a, b]$. The task of the module “planning” is to find a control strategy for determining suitable control input series within $[a, b]$ which enables the controlled system to satisfy the chosen stability criterion and to find the most suitable control input (the optimal solution) among the suitable control input series. Here it is assumed that the optimal solution is related to a quadratic cost function as

$$J = (x_d^T(k+s)Qx_d(k+s) + u_d^T(k)Ru_d(k)), \quad (3.42)$$

where x_d and u_d are the predicted system states and desired control inputs while Q and R the positive definite weighting matrices. This cost function is one kind of representation of the system energy consisting of the state energy $x_d^T(k+1)Qx_d(k+1)$ and the input energy $u_d^T(k)Ru_d(k)$. The optimal solution is therefore the one which is determined among the suitable input series related to the stability criterion and minimizes the cost function (3.42). In other word, a suitable controller should be

developed which can make x_e as an asymptotically stable equilibrium point of the controlled system (2.1) and required least energy.

The task of the module “execution” is to determine the concrete value of the optimal control input. Therefore the realization of both modules “planning and execution” are stated together in this section.

Two different strategies are considered in this thesis. They are fast exhaustive grid search method explained in Subsection 3.3.1 and inverse dynamic optimal control method stated in Subsection 3.3.2. Both of them can be applied for the module “planning and execution” to determine the optimal input solution in online process.

3.3.1 Exhaustive grid search method

Considering the system input signals separately, the bounded interval $[a_j, b_j]$ with $j = 1 \dots m$ of each input signal u_j is determined. If all bounded intervals are partitioned into several subintervals without intersections as

$$[a_j, \quad a_j + \Delta u_j, \quad a_j + 2\Delta u_j, \dots, b_j - \Delta u_j, \quad b_j] \quad (3.43)$$

with the corresponding sample interval Δu_j for each input ($\Delta u = [\Delta u_1, \dots, \Delta u_m]$ for all inputs), the number of the possible input values is finite and the method of exhaustive grid search method can be used to determine the optimal control input as stated in [SS12] and repeated briefly in the sequel.

The endpoint of each subinterval is considered as a possible value of the corresponding input signal. The cost function (3.42) and the stability of the predicted motion are calculated and evaluated for all possible values of all input signals (the possible input vector) and the corresponding predicted outputs. The possible input vector which can generate a stable behavior are stored in a set U_s . The input vector among U_s showing the smallest value of cost function is denoted as the optimal solution of the control input vector and is generated in the module “execution”. This strategy is illustrated with an example as shown in Figure 3.14

As shown in (a), the system states $\hat{x}(k+s)$ for the next prediction horizon s are predicted using the system identifier with all possible input vectors denoted in this example as $u_1 \dots u_4$. Afterwards, the stability of the motion of the closed-loop system related to the predicted system states is judged using the stability criterion as shown in (b). The input vectors u_2 and u_3 which can satisfy the chosen stability criterion are stored in the set U_s . Finally, the most suitable input among the set U_s is selected by calculating the cost function (2.1). As shown in (c), F_2 and F_3 denote the value of the corresponding cost function of u_2 and u_3 (assuming $R = 0$ here). Because $F_2 < F_3$, u_2 is determined as the new control input for the next s steps.

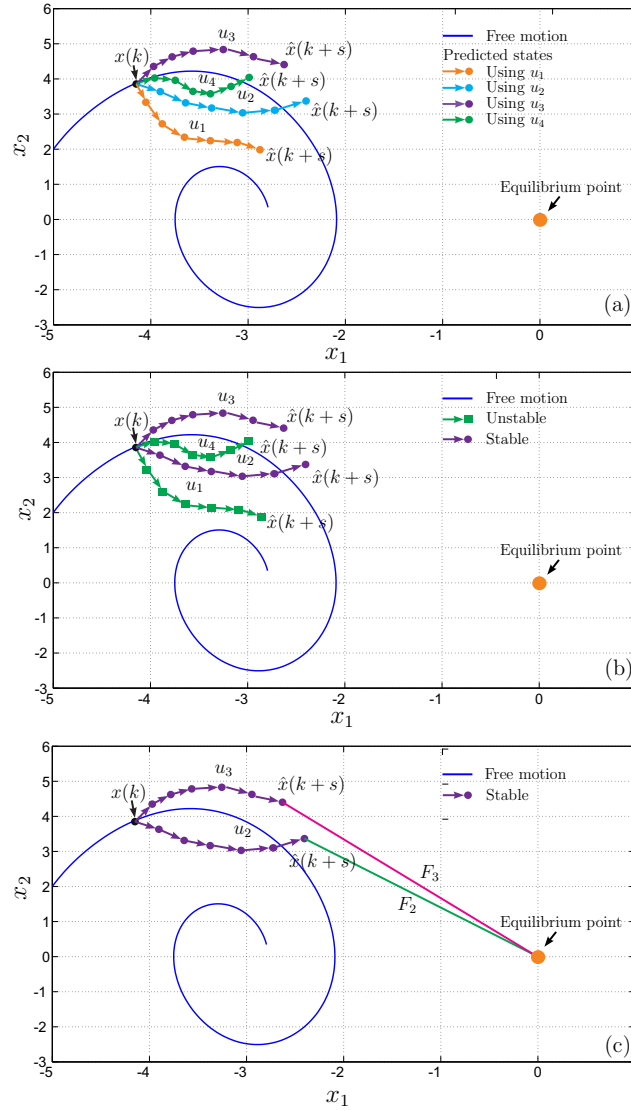


Figure 3.14: An example of the realization of modules “planning and execution” using exhaustive grid search method

Using this kind of strategy, the optimal control input can be found and the stability of the controlled system can be guaranteed. However, the computational requirement of this strategy is not optimal, because both the stability of the motion of the controlled system and the cost function are calculated for each corresponding possible input. In order to solve this problem, the use of the exhaustive grid search method is optimized as illustrated as shown with a flowchart in Figure 3.15.

After the initial training process using the system identifier, the norm of the current system states $x(k)$ is checked for each prediction horizon. In order to use the stability

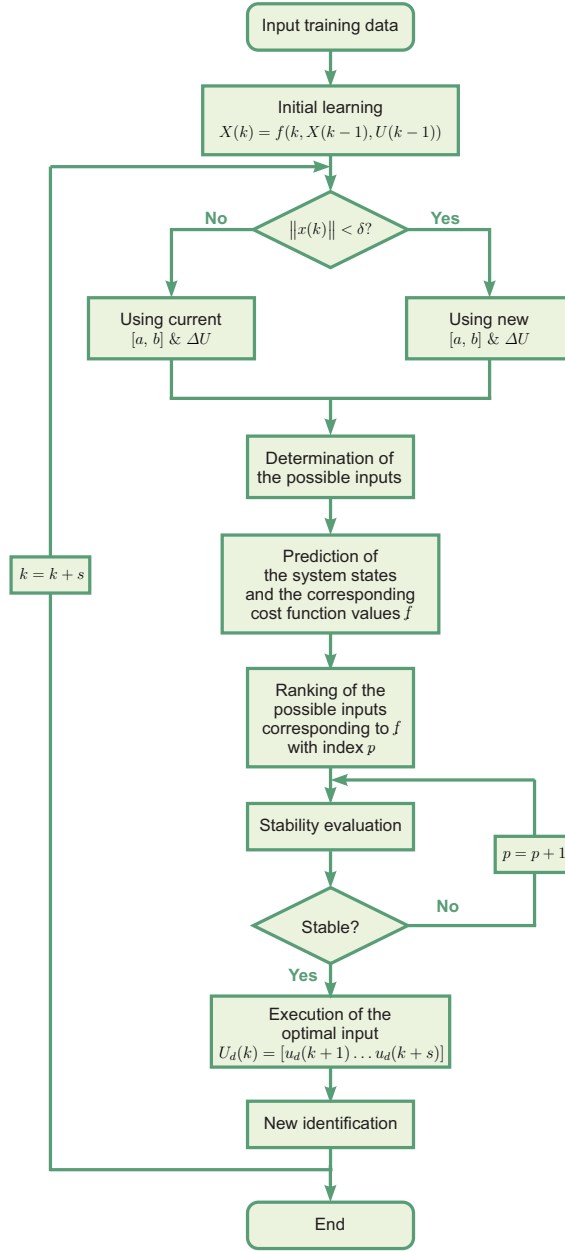


Figure 3.15: Flowchart of the cognitive stabilizer using exhaustive grid search method

criterion, a coordinate transformation with the desired equilibrium point of the controlled system as the original point is done. If $\|x(k)\|$ is larger than a predefined distance δ , which means the system states are not close to the equilibrium point of the controlled system or the absolute values of the system states are not small enough, the physical possible interval of the system inputs $[a, b]$ with a relative

great sample interval Δu are used as the elements in set U_s . As mentioned before, the system identifier can not predict the system without prediction error. If the absolute values of the real system states are much larger than the difference exciting by different possible system input vectors with small Δu , the prediction results can be similar. Therefore, there is no need to choose the possible input vectors with small sample interval in the case of $\|x(k)\| > \delta$. Furthermore, using a larger sample interval, the number of the elements in set U_s can be reduced and the suitable input vector can be found quickly. If $\|x(k)\|$ is smaller than δ , a relative small boundary $[a, b]$ and a small sample interval Δu is used. In this case, the absolute value of the system states are relative small. Therefore the accuracy of the predicted results with different system inputs with small Δu can be guaranteed. The prediction is used in order to find the most suitable input vector which can stabilized the system at the desired equilibrium point. In other words, a small boundary interval $[a, b]$ is used in this case in order to improve the computational efficiency. After the adjustment of $[a, b]$ and δ , the set U_s for the possible inputs can be determined.

Compared with the stability check, the computational requirement of the calculation of the cost function is much smaller. Therefore, the system states related to the possible input vectors are predicted using learned dynamic system behavior. A ranking of the cost function values according to the predicted system states is determined as the next step. The stability of the motion of the predicted system is checked according to the generated ranking. As long as the stability criterion is fulfilled, the stability check is stopped and the corresponding input vector is determined and executed as the optimal input vector for the next s steps. Evidently, it is necessary to learn the dynamic system behavior for each prediction horizon in order to detect the changes of the unknown system or its environment online.

Using this strategy, the computational requirements can be reduced and the relative optimal control input, which can guarantee the stability of the motion, can be planned.

3.3.2 Inverse dynamic optimal control method

If the inverse dynamics of the system can be predicted with enough accuracy, the stability criterion, which provides a range of the desired stable system states, can be used. Comparing to the exhaustive grid search method, there is no need to check the stability of predicted motion according to all possible control input separately, which leads to large computational cost. Therefore, in [NS14], a control strategy denoted as inverse dynamic optimal control method is developed.

Assuming that the inverse dynamic of the system can be identified and predicted for the given desired system states $X_d(k+1) = [x_d(k+1), \dots, x_d(k+s)]$ for each prediction horizon, the desired system input matrix can be determined as

$$u_d(k) = f(k, [X(k)X_d(k+1)], U(k)). \quad (3.44)$$

Similar to the system input values, the system states should also not exceed a certain boundary range from the practical point of view. This fact can be described as a new condition for the possible range of any desired system states $x_d(k+p)$ with $p \in [1 \dots s]$ as

$$\|x_d(k+p)\| \stackrel{!}{\leq} \|x(k+p-1)\| + \epsilon(k+p). \quad (3.45)$$

The boundary $\epsilon(k+p)$ can be determined approximately according to the previous system behavior as

$$\epsilon(k+p) = \left\| \begin{bmatrix} \|x_1\|_\infty & \|x_2\|_\infty & \cdots & \|x_n\|_\infty \end{bmatrix} \right\| \quad (3.46)$$

with $x_n = [x_n(1) \ x_n(2) \ \cdots \ x_n(k+p)]$.

The optimal desired system inputs are determined by minimizing the cost function $u_d(k+p-1) = \operatorname{argmin} J$ using the active set algorithm [GMW81] with the inequality constraints (3.45) and suitable stability criterion. This control strategy is illustrated in Figure 3.16.

After the initial training process using the chosen system identifier for the inverse model of the system, the norm of the current system states $\|x(k)\|$ is also firstly checked for each prediction horizon similarly like using the exhaustive grid search method. A coordinate transformation with the desired equilibrium point of the controlled system as original point is also needed in the whole process as the requirement of the stability criterion. If $\|x(k)\|$ is larger than a predefined distance δ , the stable range $[\alpha \ \beta]$ of the desired system states are determined using the stability criterion according to the equation (3.45) for the next step. The optimal desired system input vector according to the desired system state vector among $[\alpha \ \beta]$ using the identified system model for the next step is determined by minimizing the cost function. Repeating iteratively this process p times, the optimal desired system input matrix can be determined for the next s steps. If $\|x(k)\|$ is smaller than δ , an integral controller is applied. The difficulty to predict the system dynamics with a high accuracy results from small absolute values of the system transformed states and the optimal desired input vector searched among the whole physical possible range $[a \ b]$. Therefore an integral controller (I-controller) with a certain gain vector K_I is applied due to its advantages of eliminating static control error. Using this strategy, the optimal control inputs $U_d(k) = [u_d(k) \dots u_d(k+s-1)]$ for the prediction horizon can be executed. Evidently, a learning process for each prediction horizon is also required here.

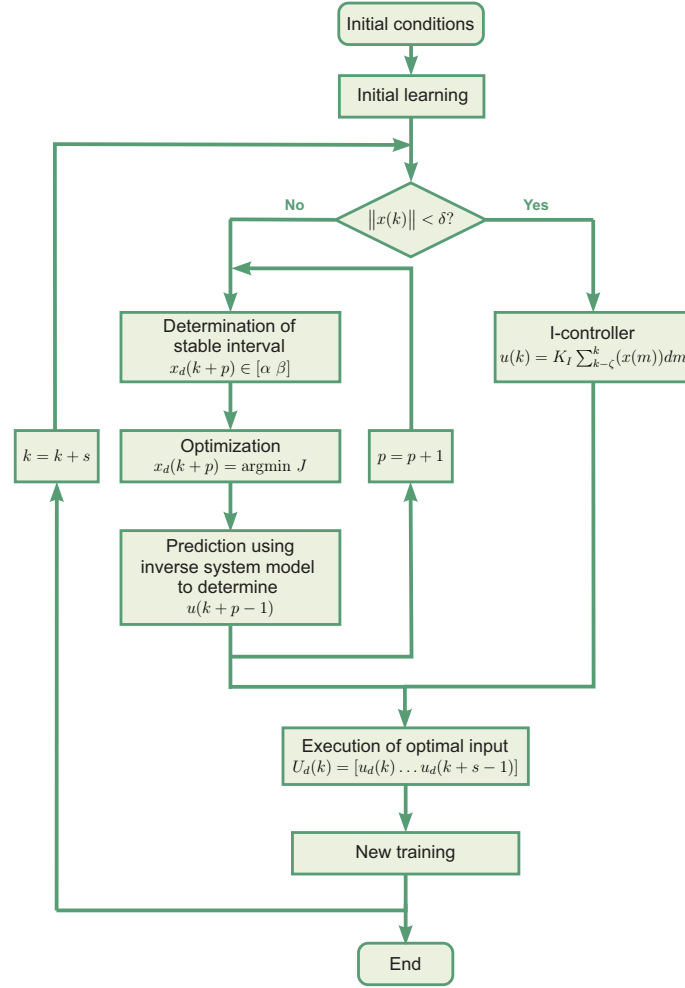


Figure 3.16: Flowchart of the cognitive stabilizer using direct input optimization with inverse model

3.4 Realization of the whole framework

As mentioned before, each module of the cognition-based framework should be realized separately at first and then combined with autonomous communication in order to realize the whole framework. The main elements of the modules in the cognitive framework are the system identifier, the stability criterion, and the strategy generator. They can be realized with different methods as explained in the previous sections. In order to discuss it clearly, an overview for the realizations of the whole framework is shown in Figure 3.17.

Some suitable methods for each main element are shown in this figure. The possible combinations of the realization of the whole system are also denoted here with the combination lines.

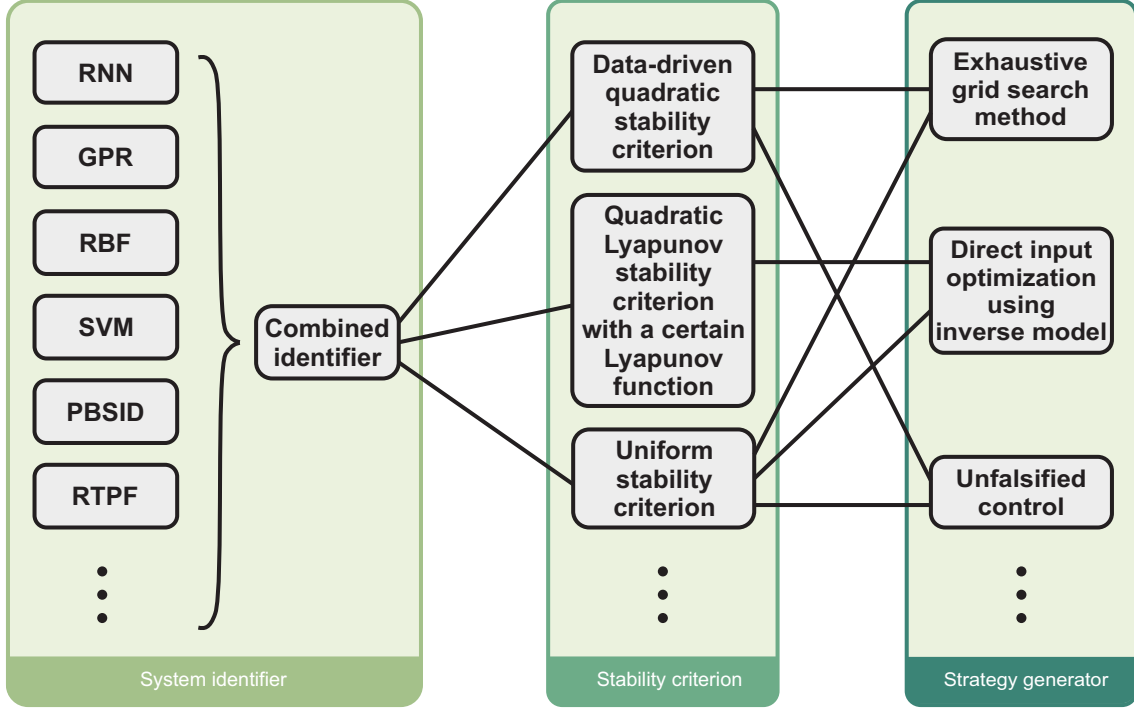


Figure 3.17: Overview of the realization of the whole framework [NS15]

As stated in Section 3.1, various kinds of methods can be used to identify and predict the system dynamics of unknown nonlinear MIMO systems with the I/O measurements online. The possible methods RNN, GPR, RBF, SVM, PBSID, and RTPF etc. can be used separately as the system identifier. In order to combine the advantages of them and apply the system identifier for most kinds of system, the combined identifier should be designed based on the existing identification methods with a suitable tuning algorithm. Therefore the combined identifier is always applied as the suitable system identifier for the whole framework.

Except the data-driven quadratic stability criterion, quadratic Lyapunov stability criterion with a certain Lyapunov function, and uniform stability of switched systems, there are also other suitable stability criteria for the cognitive framework, which are not considered in this thesis. For different stabilization task or requirement, different stability criterion is needed. For example, the uniform stability of switched systems is considered as the suitable stability criterion for some kinds of systems with physical bounded states. The reason for it is that the uniform stability can supervise with less computational load how to design the control strategy that can fulfill the stabilization task. If the inverse model of a system without known physical bounded states can be identified and predicted using combined identifier, the quadratic Lyapunov stability criterion with a certain Lyapunov function can be considered as the suitable stability criterion, because the strategy generator can be

realized using inverse dynamic optimal control method which is more efficient than the exhaustive grid search method.

The task of the strategy generator is to determine the concrete values of the optimal control input series for the next s steps. Different control strategies for example exhaustive grid search method, inverse dynamic optimal control method, and unfalsified control method (if it can be extended for MIMO systems) etc. can be applied here. All of them are able to find the suitable input series related to the suitable stability criterion and can minimize the cost function without additionally knowledge of the system.

For the realization of the whole cognitive framework, there are different realization combinations of the possible methods for each module. For example the combined identifier, data-driven quadratic stability criterion, and exhaustive grid search method is one of the possible combination for the realization of the whole cognitive framework. Other possibilities are also given in Figure 3.17.

For different systems, the control performance using different realization of the cognitive framework can be different. The cognitive stabilizer should also be able to choose the most suitable realization autonomously according to the I/O measurement and the control goal given by the user. The autonomous choosing algorithm of the whole framework is developed in this thesis as shown in the following based on the methods explained in the Sections 3.1, 3.2, and 3.3:

- The system model and its inverse model are identified and predicted using the combined identifier with initial training as well as the test I/O measurement. The suitable parameters of the combined identifier (e.g. the covariance function and k_c) are determined at first according to the evaluation of the identification and prediction results. It should be mentioned that not all inverse model of unknown systems can be identified and predicted using the combined identifier with acceptable accuracy. The possibility of the identification of the inverse model is stored in the cognitive framework after the initial training process.
- The user gives a rank of the stability criteria with respect to the requirement of the stabilization task. Logically, if the inverse model of the system can not be identified with acceptable accuracy, the quadratic Lyapunov stability criterion with a certain Lyapunov function is not considered more. If the system states are not bounded within a known physical interval, the uniform stability of switched systems is also not considered more.
- According to the rank of the stability criteria and the combination possibilities of the whole framework, suitable control strategy can be chosen with respect to the required computational load. Considering the realizations of the control strategy in this thesis, inverse dynamic optimal control method requires less computational load than exhaustive grid search method.

Using this algorithm, the most suitable realization of the whole framework can be determined autonomously according to the requirement of the user. The choosing algorithm should be extended furthermore if other kind of realizations are considered in the further.

4 Simulation Results of Applications using Cognitive Stabilizer

In order to check for the correctness and the performance of the proposed cognitive stabilizer, it is applied to some simulation examples which are outlined discussed in this Chapter. In Section 4.1, the simulation examples are introduced with their mathematical model. In Section 4.2, the autonomous selection of the suitable realization of the cognition-based framework for the cognitive stabilizer applied in the examples is explained. The stabilization results using the suitable realization of the proposed cognitive stabilizer are shown and analyzed.

4.1 Introduction of the simulation examples

Two simulation examples are considered in this thesis: pendulum system and Lorenz-system. Both of them are benchmark nonlinear systems for the control technique.

4.1.1 Pendulum system

The first simulation example is the pendulum system which is a nonlinear dynamical single-input-multi-output (SIMO) system as shown in Figure 4.1.

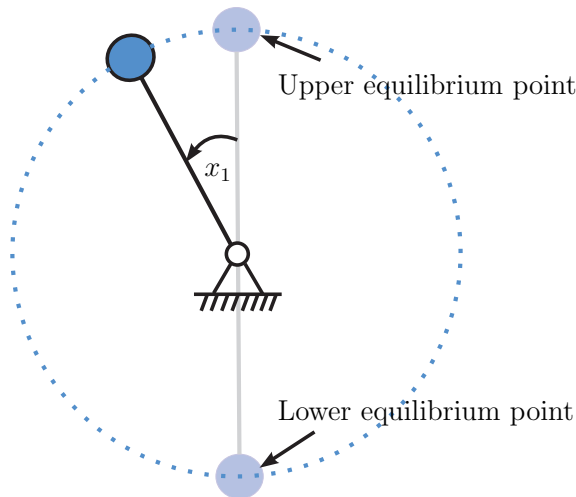


Figure 4.1: Pendulum system

The mathematical model of the pendulum system is given with its state space representation as

$$\begin{cases} \dot{x}_1(t) = x_2(t) \\ \dot{x}_2(t) = -10\sin(x_1(t) - \pi) - x_2(t) + u(t), x(t=0) = x_0, \end{cases} \quad (4.1)$$

where x_1 and x_2 are the measured system states which represent the angular and the angular velocity of the pendulum and u is the single control input.

The pendulum system has two equilibrium points as shown in the Figure 4.1: the upper equilibrium point which is an unstable equilibrium point and the lower equilibrium point which is stable. The control task is to stabilize the pendulum system at its arbitrary equilibrium point. It is assumed that only the system states x_1 and x_2 , which are also the measured system outputs, as well as the control input and control goal are known for the controller.

4.1.2 Lorenz-system

The second simulation example is the Lorenz-system which can be described as a chaotic nonlinear dynamical MIMO system. The mathematical model of the Lorenz-system is described by

$$\dot{x} = \begin{pmatrix} -\sigma x_1 + \sigma x_2 + u_1 \\ -x_1 x_3 - x_2 + r x_1 + u_2 \\ x_1 x_2 - b x_3 \end{pmatrix}, \quad x(t=0) = x_0, \quad (4.2)$$

where x_1 , x_2 , and x_3 are the measured states as well as u_1 and u_2 are the control inputs. The value of the parameters are given for the simulation as $\sigma = 10$, $r = 28$, and $b = \frac{8}{3}$. This system has three equilibrium points:

- $P_1 = (0, 0, 0)$,
- $P_2 = (\sqrt{b(r-1)}, \sqrt{b(r-1)}, r-1) = (8.4853, 8.4853, 27)$, and
- $P_3 = (-\sqrt{b(r-1)}, -\sqrt{b(r-1)}, r-1) = (-8.4853, -8.4853, 27)$.

The free motion of the Lorenz-system is shown in the Figure 4.2.

It can be seen from the free motion, that the dynamic of the Lorenz-system resembles a butterfly or figure eight. The system can not be stabilized at any equilibrium point of it without controller. The desired control task in this thesis is to stabilize the Lorenz-system at an arbitrary equilibrium point of it. It is assumed that the system states x_1 , x_2 , and x_3 , which are also the measured system outputs, as well as the control input and control goal are known for the controller.

4.2 Simulation results

As mentioned in Section 3.4, different realizations of the cognitive stabilizer can be applied to the same system. In this section, the reason of selecting of different realizations, the simulation results and the analysis of the performance of the proposed cognitive stabilizer are stated for the pendulum system and Lorenz-system separately.

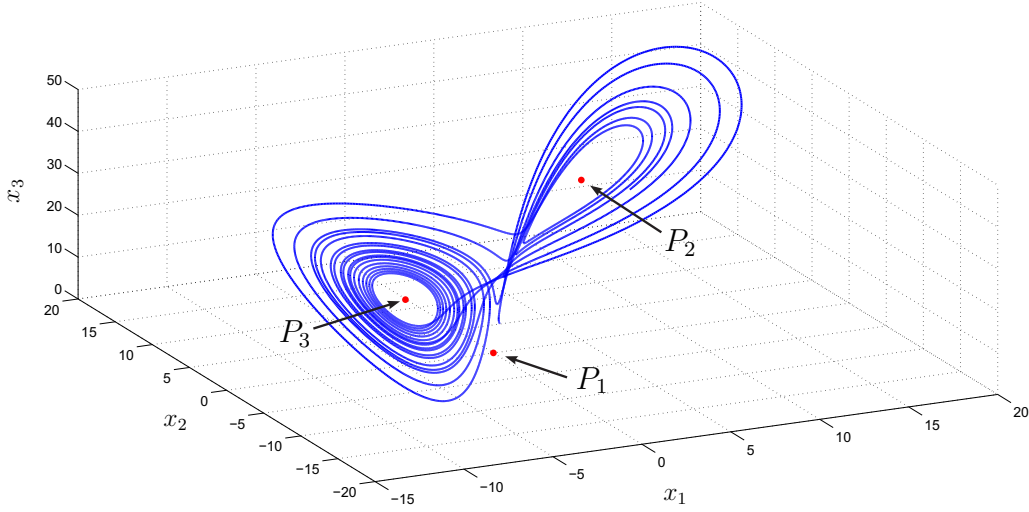


Figure 4.2: Free motion of Lorenz-system

4.2.1 Simulation results of the pendulum system

As stated in Section 3.4, in order to choose the suitable realization of the cognitive stabilizer, the system model and its inverse model should be first identified separately using the combined identifier with initial training I/O measurement. Using the identified models, the system dynamics and inverse dynamics are predicted using test I/O measurement. The prediction result is used to evaluate whether the system model and the system inverse model can be identified and predicted with an acceptable accuracy, so that the corresponding realization can be used for the cognitive stabilizer applied to the pendulum system.

The identification and prediction of the system model are tested with the simulation at first. The parameters of this test are defined here as training period $l = 500$, the sample time $Ts = 10^{-4}s$, the initial system states $x_0 = [-3 \ 5]$, $k_c = 1400$, $s = 20$, $\delta = 5$, and $u \in [-50 \ 50]$. For this simulation example, the prediction error threshold is predefined by the user as $AE = \pm 0.5$, $MSE = 0.1$, and $ARE = 0.1$. The corresponding prediction results are shown in Figure 4.3 with the prediction errors E_1 and E_2 in Figure 4.4.

In the training process, real I/O data denoted with black lines as shown in Figure 4.3 are given to the combined identifier for the first 500 steps. The training input here is given with the normal distributed random signal. The system outputs for $k = 500$ to $k = 1500$ are predicted with given test input data (with sequence step signal). This input data and the corresponding test output data are simulated real I/O data which are also denoted with black lines. As introduced in Section 3.1.3, a suitable identification method between NARX-RNN and GPR is tuned automatically by

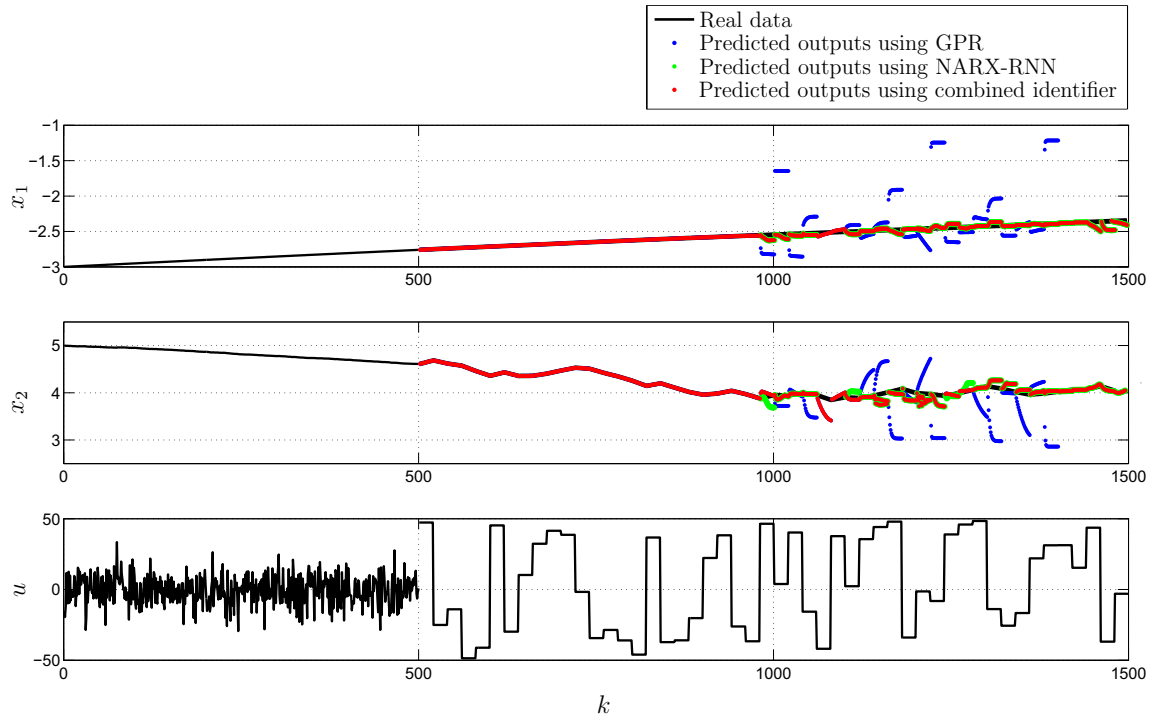


Figure 4.3: Prediction results of the normal model of the pendulum system using combined identifier

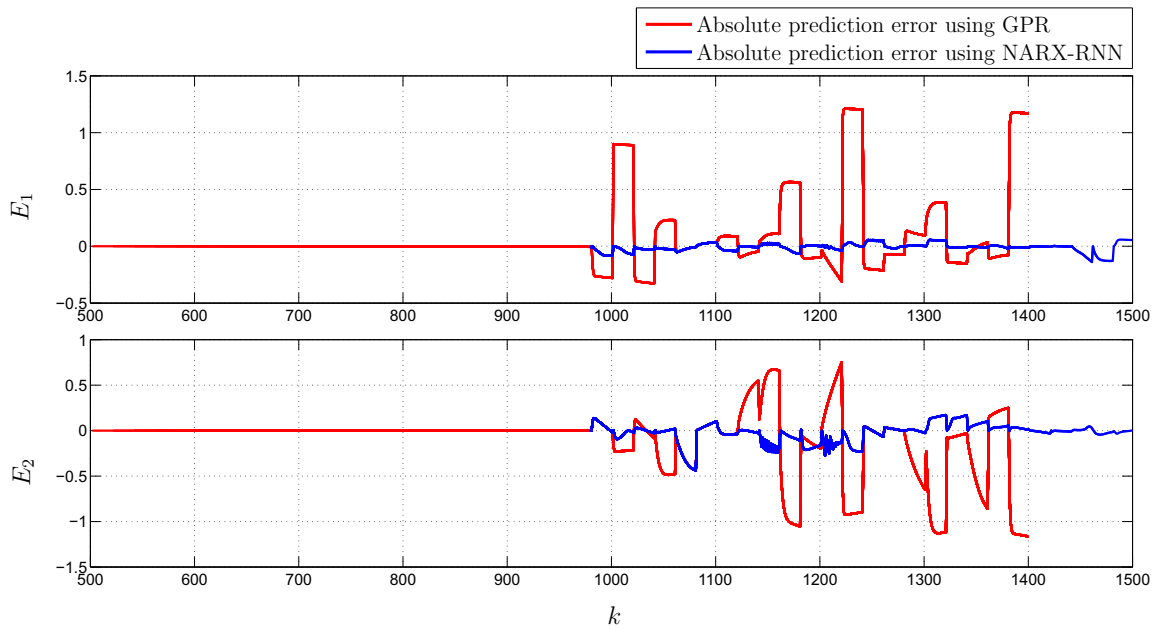


Figure 4.4: Prediction error of the normal model of the pendulum system using combined identifier

the combined identifier for the same system in different cases. In order to show this automatic tuning clearly, the predicted system outputs are shown in Figure 4.3 with blue points for using GPR, with green points for using NARX-RNN, and with red points for the final prediction result using combined identifier. It can be seen that for the first 500 steps in the prediction process ($k = 500$ to $k = 1000$), GPR can predict the system outputs with high accuracy. From $k = 1000$ to $k = 1400$, although GPR is predefined to do the prediction task, the prediction results are not more acceptable detected by the validation process of the combined identifier automatically. Therefore NARX-RNN is used to improve prediction results. After 1400 steps, NARX-RNN is used and can give accepted predicted outputs. The strategy of the autonomous tuning of the combined identifier is also shown, with the absolute prediction errors, in Figure 4.4. The prediction error using GPR is denoted with red lines and for NARX-RNN blue lines. It can be analyzed from this result that the parameter k_c can be set as 1000 later for the online control process. The final prediction results denoted with red points as shown in Figure 4.3 are always the more suitable one between the prediction result using GPR and NARX-RNN.

A corresponding error analyze of the prediction error using combined identifier is given in Table 4.1.

Table 4.1: Prediction errors			
State	AE	MSE	ARE
x_1	-0.0055	0.0008	0.0038
x_2	-0.0126	0.0060	0.0059

It can be concluded that the model of the pendulum system can be predicted using the combined identifier with acceptable accuracy in the whole process, because all the errors (AE, MSE, and ARE) are smaller than the predefined error threshold by the user. As a result, the exhaustive grid search method, which requires the predicted system dynamics, can be used to realize the module “planning” as shown in the overview of the realization of the whole cognitive framework in Figure 3.17.

In order to check whether the other stated method (direct input optimization using inverse model), which requires the identified inverse system model, for realizing the module “planning” can also be used for stabilizing the pendulum system, the identification and prediction capability of the combined identifier is evaluated for the inverse model of the pendulum system. Different from the evaluation of the prediction capability for system outputs, ARE is considered as the only criterion for the acceptance of the system input prediction, because the error thresholds of AE and MSE are difficult to be given by the user if the input signal is a random or sequence step signal with not small absolute values. For the pendulum system, the acceptable error threshold for ARE of the identified and predicted input is given

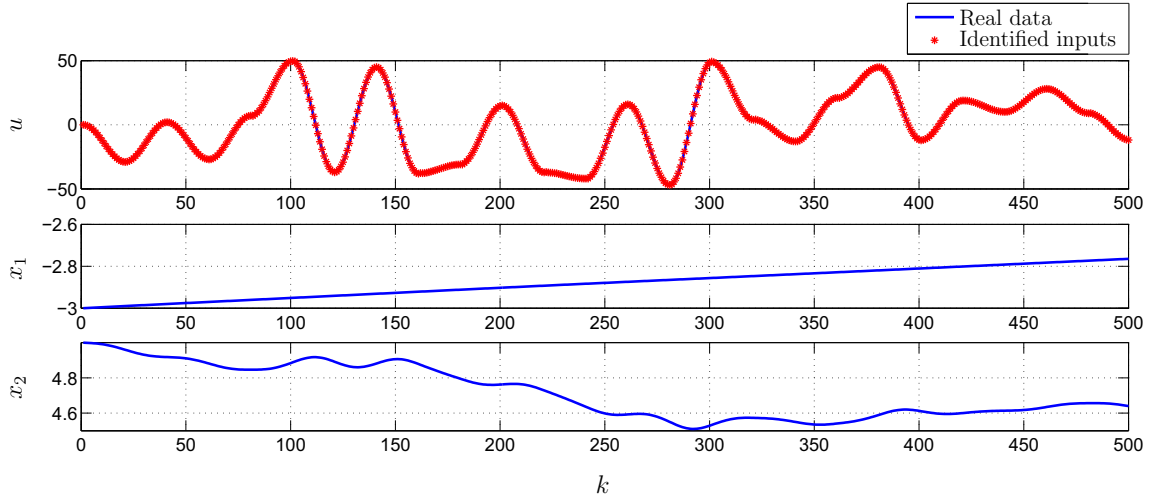


Figure 4.5: Identification results of the inverse model of the pendulum system using combined identifier

with the value 0.1. The identification result is shown in Figure 4.5 at first to check the correctness of the identified inverse model of the pendulum system.

The training real I/O data are denoted with blue lines. It can be seen clearly that the identified system input denoted with red points is almost the same as the real input data. The corresponding identification error ARE is 0.0325 which is smaller than the acceptable error threshold 0.1. So the identified inverse model of the pendulum system can be used to calculate the system input with given desired system outputs. This is tested in the following and the corresponding result is shown in Figure 4.6.

The desired outputs for the next $s = 20$ steps are given and denoted with cyan lines. The inputs are determined using the identified inverse model and shown with red lines. In order to check the correctness of the determined input values, the real system outputs using these inputs are simulated and shown with blue points. It can be seen that for the first 10 steps, the blue points can follow the desired outputs very well. In order to check the performance of the determination of the suitable input for the whole 20 steps, the error between the desired outputs and the real outputs is analyzed and shown in Table 4.4. Here, the tracking error threshold is predefined by the user as $AE = \pm 0.5$, $MSE = 0.1$, and $ARE = 0.1$.

Table 4.2: Tracking errors

State	AE	MSE	ARE
x_1	-0.0039	0.0001	0.0014
x_2	-0.0410	0.0029	0.0088

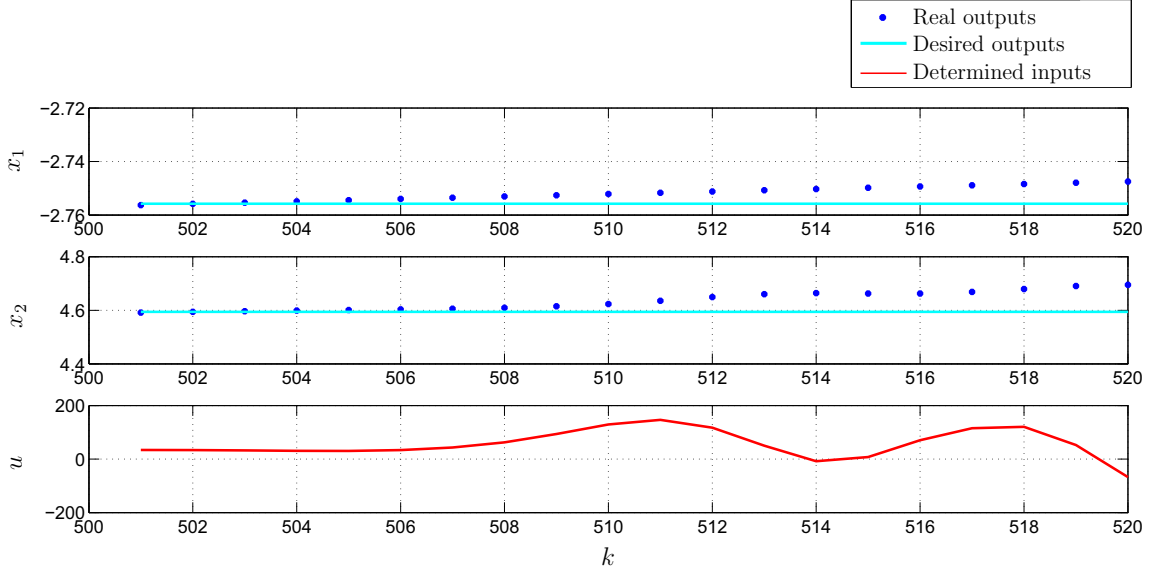


Figure 4.6: Tracking results of the inverse model of the pendulum system using combined identifier

It can be concluded from the table that all tracking errors are smaller than the predefined error threshold. As a result, the inverse model of the pendulum system can be identified and applied to determine new system input with given desired output using combined identifier. Therefore, the method direct input optimization using inverse model can be used for the realization of the module “planning”.

The suitable realization of the module “expert knowledge” should be selected according to the performance of the system and the expectation of the user as the second step. Because both system states are not bounded in a fix interval, the uniform stability of switched systems is not considered as the suitable stability criterion for this case. By stabilizing the pendulum system it is desired that the pendulum should be stabilized at best directly to its upper equilibrium point without large and repeatedly pendulum around the equilibrium point. Using quadratic Lyapunov stability criterion with a certain Lyapunov function, the condition (3.37) should always be fulfilled. Therefore, the system states (angle position and angle velocity) should have the same weighting. Obviously, if the angle velocity is reduced faster than the angle position, more input energy is needed to stabilize the pendulum system at its upper equilibrium point directly. As mentioned before, the input is bounded in a fest interval $u \in [-50 \ 50]$. So the required input energy may not be provided in the real system. As a result, the quadratic Lyapunov stability criterion with a certain Lyapunov function is also not used here. There is no constraint of using data-driven quadratic stability criterion for the pendulum system, therefore it is selected as the suitable stability criterion for the module “expert knowledge”.

The realization of the whole cognitive framework can be selected now according to the overview as shown in Figure 3.17: learning and predicting the system dynamic behavior using combined identifier, judging the quadratic stability of the predicted system motion using data-driven quadratic stability criterion, and finding the optimal control input vector using exhaustive grid search method. This kind of realization is denoted as the first kind of realization in this thesis.

Applying the stated realization above to the pendulum system, the stabilization result is shown in Figure 4.7. The initial state $x_0 = [-3, 4]$ is taken in this example.

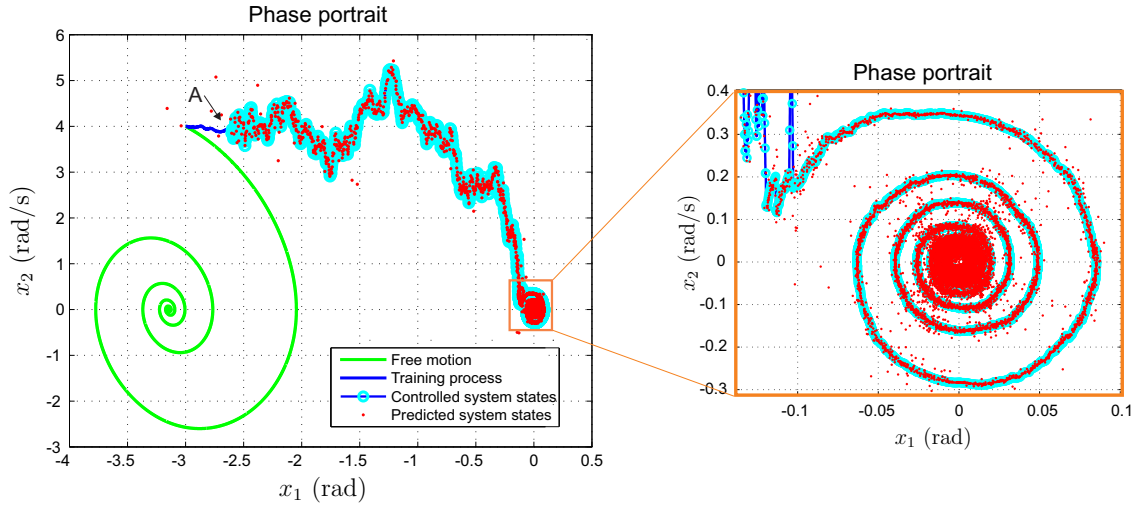


Figure 4.7: Phase portrait of the controlled pendulum system at its upper equilibrium point

The origin point of the phase portrait denotes the upper equilibrium point. The green line indicates the free motion of the pendulum system. It becomes clear that the system will not go to the upper equilibrium point without control input. The red points indicate the predicted system states by the combined identifier. At the end of the training period (here denoted from the initial point to point A with blue line), the combined identifier has been trained to have the capability to predict the real system with acceptable accuracy. The control strategy is applied after the point A. The cyan points denote the actual closed-loop response of the system approaching the origin point using the proposed realization of cognitive stabilizer. The corresponding time history of each system output and the control input are shown in Figure 4.8.

It can be seen, after about $k = 4000$ (0.4 seconds), the position and the velocity of the system almost converge to zero. In order to check the quadratic stability of the controlled pendulum system during the whole control process, Figure 4.9 shows the convex conic cone C consisting of the transmitted system states w_1 and w_2 using the map mentioned in Section 3.2.1.

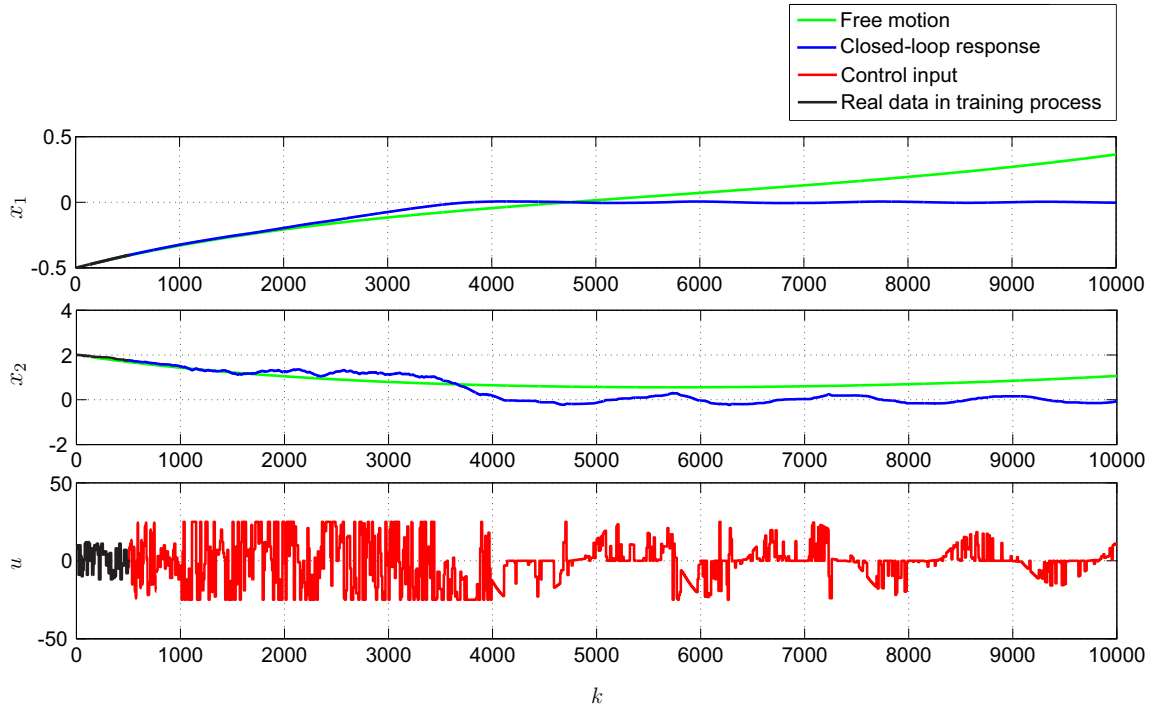


Figure 4.8: Time history of the outputs and input of the controlled pendulum system (upper equilibrium point)

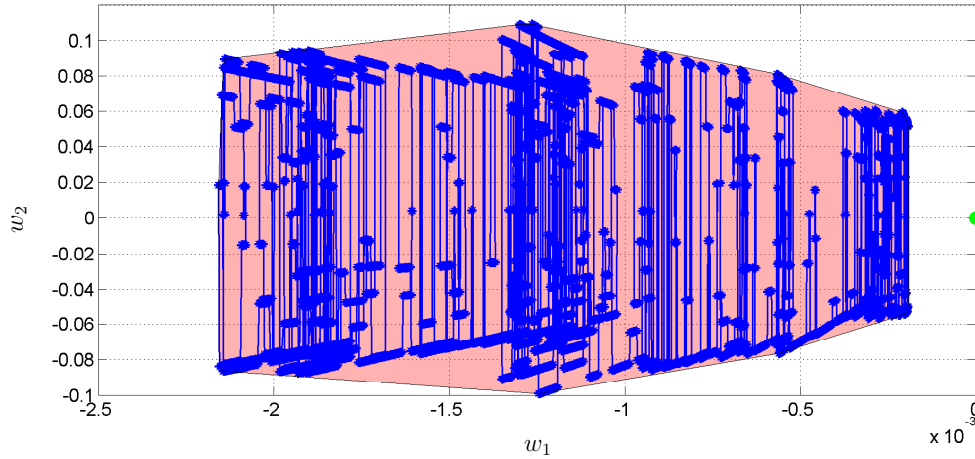


Figure 4.9: Convex conic cone of the transformed system states of the pendulum

The cone denoted with red area is located in the negative half space, which means that the system control input can indeed make the system satisfy the requirement of the system quadratic stability during the whole control process.

In order to check the correctness of the proposed cognitive stabilizer, two couples of symmetrical initial points such as $[-0.5 \ 2]$ and $[0.5 \ -2]$, $[0.5 \ 2]$ and $[-0.5 \ -2]$ are tested for the case of upper equilibrium point in the following. The pendulum system has symmetrical dynamic behavior, which means that its dynamic behavior should still be symmetrical using the same control strategy and symmetrical initial conditions. The simulation results are shown in Figure 4.10.

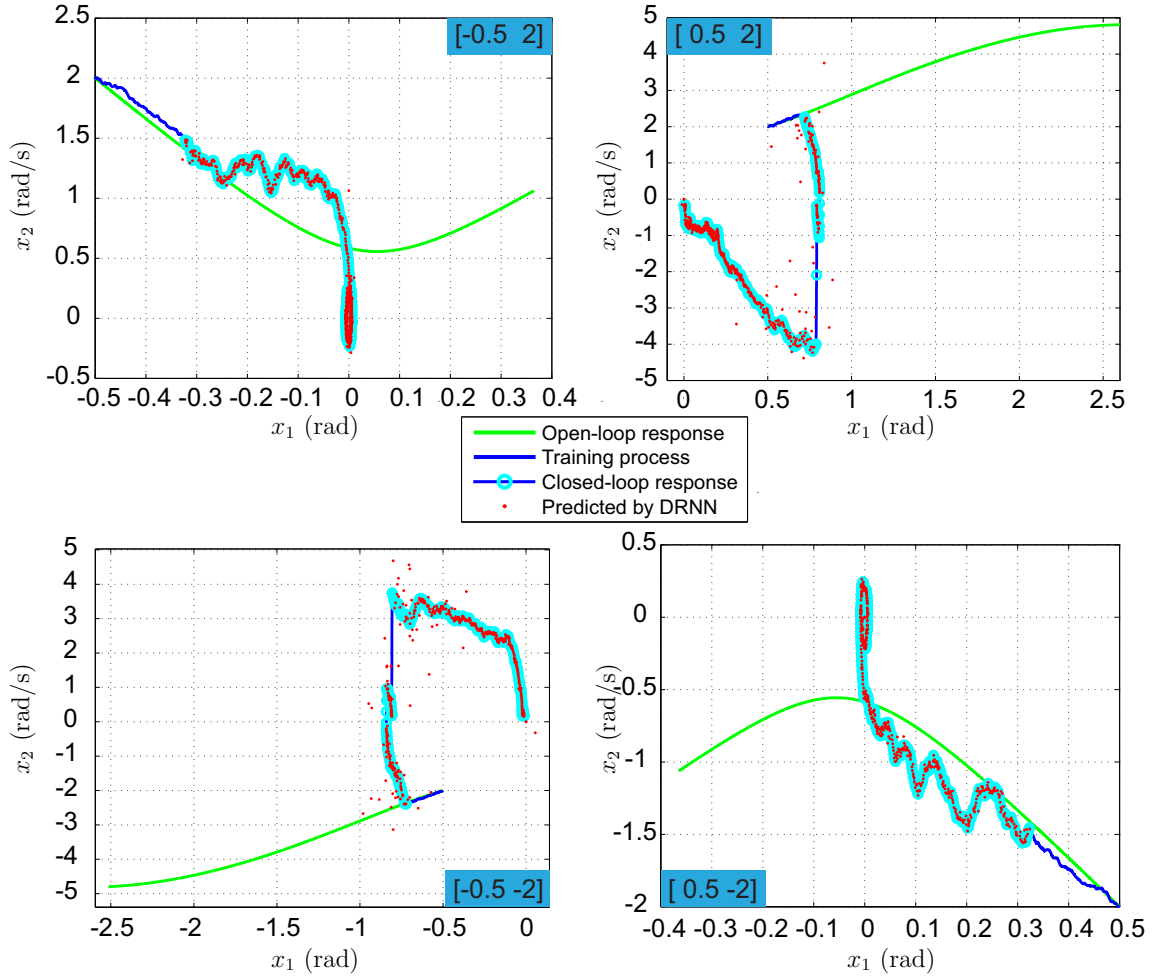


Figure 4.10: Phase portrait of the controlled pendulum system at its upper equilibrium point with symmetrical initial conditions

It can be seen that the pendulum system can be stabilized at its upper equilibrium point with the given different initial conditions. The corresponding closed-loop responses with $x_0 = [-0.5 \ 2]$ and $x_0 = [0.5 \ -2]$ are similar as well as with $x_0 = [0.5 \ 2]$ and $x_0 = [-0.5 \ -2]$. It is pointed out that the simulation results are not totally symmetric, because the model of the unknown system is identified using combined identifier which can represent the system model similarly but not totally identical.

It can be concluded that the cognitive stabilizer realized using the first kind of realization can stabilize the pendulum system at its upper equilibrium.

4.2.2 Simulation results of the Lorenz-system

Similar as the control process of stabilizing the pendulum system, the combined identifier should first be tested with some test I/O data of the Lorenz-system in order to determine the parameters of the combined identifier as well as to determine the realization methods of the cognitive stabilizer. The prediction result is already stated as an example of the combined identifier in Section 3.1.3 with the Figures 3.11 and 3.12. It can be concluded that the combined identifier can predict the dynamics of the Lorenz-system with acceptable accuracy. The parameters used for the prediction are $T_s = 10^{-3}\text{s}$, $T_t = 0.5\text{s}$, $\eta = 30$, $\alpha = 0$, $s_n = 0.05$, $k_c = 600$, $x_0 = [-10 \ 10 \ 25]$, and $s = 5$.

The combined identifier is also applied to identify the inverse Lorenz-system model. The identification process is first tested and the corresponding results are shown in Figure 4.11. The parameters used here are $T_s = 10^{-4}\text{s}$, $T_t = 0.05\text{s}$, $x_0 = [-10 \ 10 \ 25]$, and $k_c = 0$. The real inputs are bounded in the interval $[-2000, 2000]$.

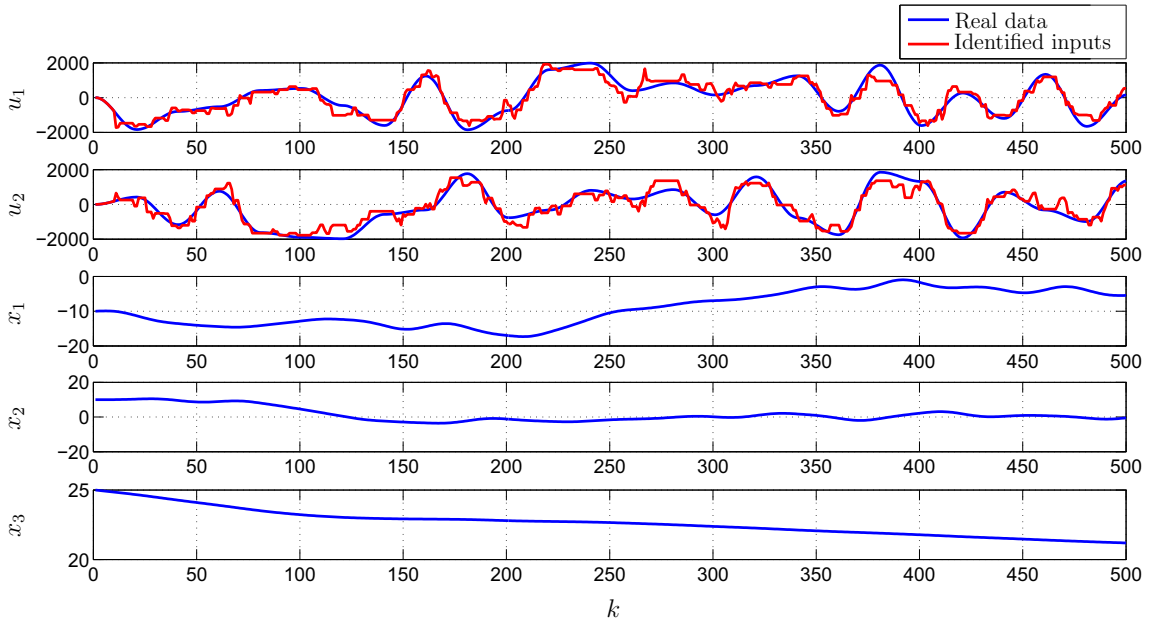


Figure 4.11: Identification results of the inverse model of the Lorenz-system using combined identifier

The blue lines denote the training inputs and outputs data of the simulated real system. Different from the identification of the normal model of the Lorenz-system,

the inputs of the system should be identified for the inverse model of the Lorenz-system. The identification error threshold is predefined as 0.1. Using the combined identifier, the inputs can be identified with acceptable accuracy as shown with red lines and the error analyze given in Table 4.3.

Table 4.3: Prediction errors

Input	ARE
u_1	0.0836
u_2	0.0074

It can be seen that the ARE for both inputs are small enough (as stated before, the acceptable maximum value of ARE is $10\% = 0.1$). As a result, it can be concluded that the inverse model of Lorenz-system can be identified with acceptable accuracy using combined identifier. Applying this identified inverse model to calculate the new system inputs according to the given new desired system outputs, the corresponding result is shown in Figure 4.12.

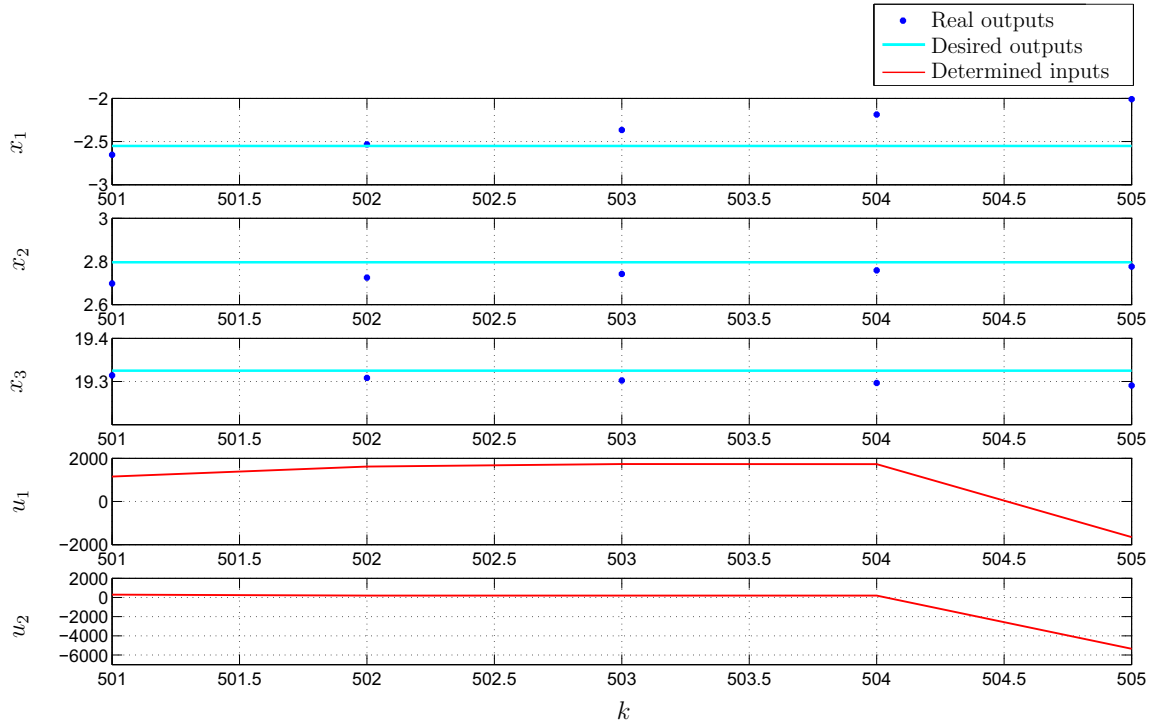


Figure 4.12: Tracking results of the inverse model of the Lorenz-system using combined identifier

The desired outputs for the next $s = 5$ steps are given and denoted with cyan lines. The inputs are determined using the identified inverse model and shown with red

lines. In order to check the correctness of the determined input values, the real system outputs using these inputs are simulated and shown with blue points. The errors between the desired outputs and the real outputs are analyzed and shown in Table 4.4. Here, the prediction error threshold is also predefined as $AE=0.5$, $MSE=0.1$, and $ARE=0.1$.

Table 4.4: Prediction errors			
State	AE	MSE	ARE
x_1	-0.2020	0.0945	0.0969
x_2	0.0564	0.0039	0.0207
x_3	0.0228	0.0006	0.0012

The values of AE, MSE, and ARE for all three system states are smaller than the error threshold. Therefore, the combined identifier can be used for determining the inputs for the next s steps according to the desired outputs.

As a result, both the exhaustive grid search method and the direct input optimization using inverse model can be used to realize the module “planning”. Because the physical boundary of the three system states are not known, the uniform stability of switched systems is not considered as the suitable stability criterion for this case. There are no special requirement given by the user, therefore both quadratic Lyapunov stability criterion with a certain Lyapunov function and data-driven quadratic stability criterion can be used as the stability criterion in the module “expert knowledge” for the Lorenz-system. As stated before, it needs more computational load than the quadratic Lyapunov stability criterion with a certain Lyapunov function. Therefore, the most suitable realization of the cognitive stabilizer for Lorenz-system is selected automatically as the combination of the combined identifier, the quadratic Lyapunov stability criterion with a certain Lyapunov function, and the inverse dynamic optimal control method according to the realization of the whole cognitive framework as shown in Figure 3.17. The corresponding control result using this kind of realization (denoted as the second kind of realization) are illustrated in the following.

The desired control task is to stabilize the Lorenz-system at one of its equilibrium points. The second equilibrium point P_2 is considered at first. Afterwards, the Lorenz-system should be stabilized from P_2 to another equilibrium point P_3 , in order to show the performance of the proposed cognitive stabilizer. With the conditions of the sample time $Ts = 10^{-4}s$, the initial system states $x_0 = [-20 \ -15 \ 25]$, $s = 20$, $h = 100$, unit matrices Q and R , $u \in [-1500 \ 1500]$, and $l = 500$, the corresponding results are shown with the time history of both system states and inputs in Figure 4.13 as well as with the phase portrait in Figure 4.14.

During the training period (here denoted with black line), the inverse model of the Lorenz-system is identified using the combined identifier. After the training

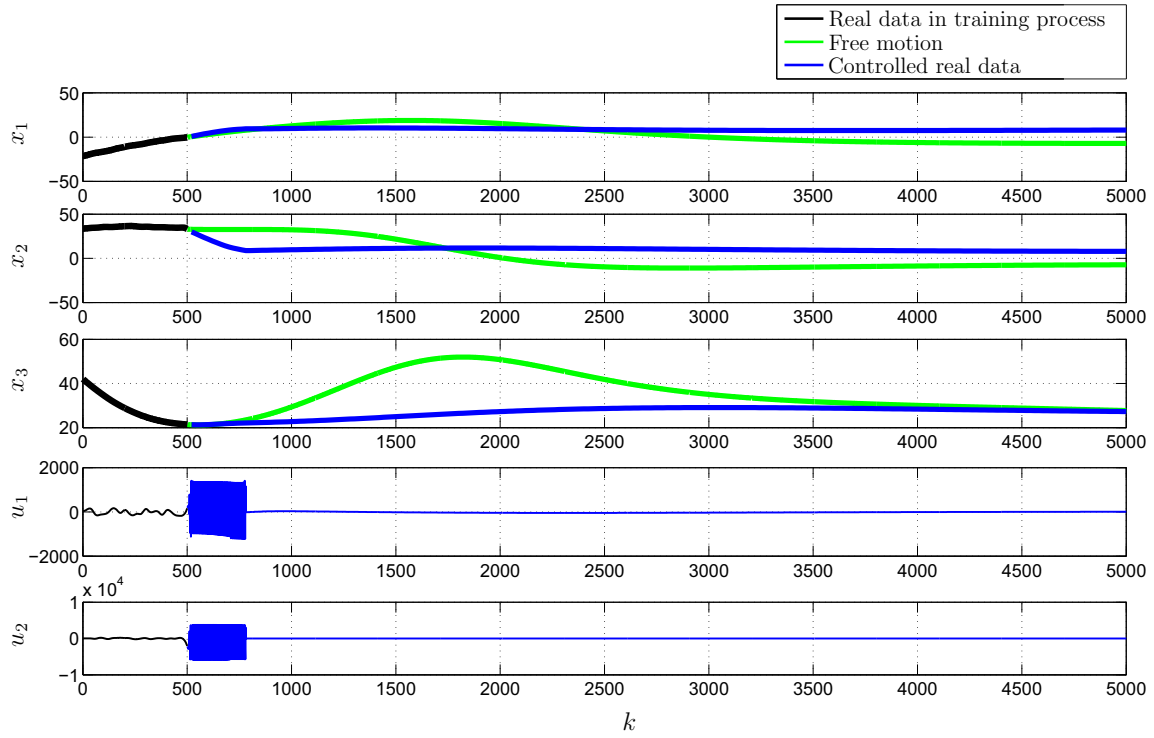


Figure 4.13: Time history of the controlled Lorenz-system using the second kind of realization of the cognitive framework (arbitrary point $\rightarrow P_2$)

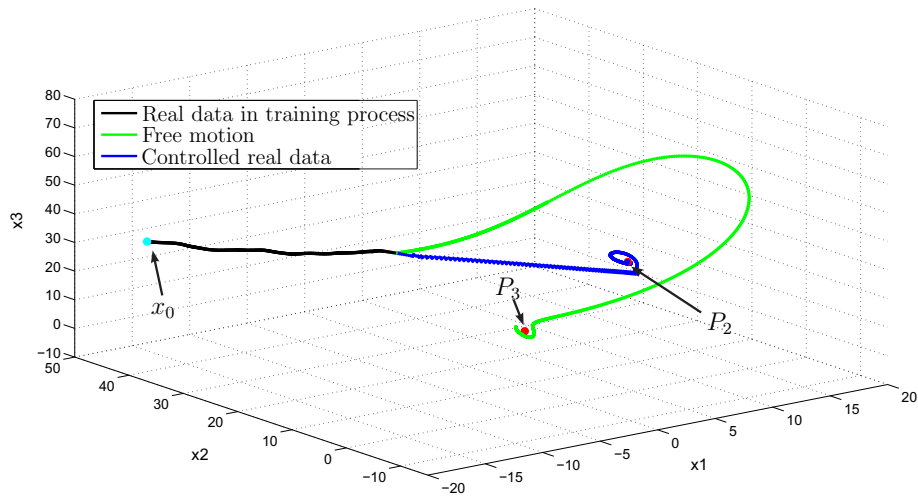


Figure 4.14: Phase portrait of the controlled system using the second kind of realization of the cognitive framework (arbitrary point $\rightarrow P_2$)

period, the quadratic Lyapunov stability criterion with a certain Lyapunov function is used to give the suitable range of the desired system states x_d with respect to the stability of the switched system for the next s steps. The new system inputs are calculated according to the identified inverse system model and x_d . Applying these new inputs as the control input to the real system and repeating this process (training, determining and applying the new inputs) for each s steps, the real closed-loop response of the system denoted by the blue lines can approach an area near the desired equilibrium point P_2 after about 250 steps (0.025 seconds) as shown in Figure 4.13. After 0.025 seconds, an I controller is used to approach P_2 , of which closed-loop response can be seen clearly in both Figures 4.13 and 4.14. The green line indicates the free motion of the Lorenz-system. It becomes clear that the system will not go to any of its equilibrium point without control input.

In order to check the performance of the proposed cognitive stabilizer further, the Lorenz-system will be stabilized directly from the equilibrium point P_2 to P_3 starting at $t = 1$ s. The corresponding simulation results is shown in Figure 4.15 and in the phase portrait in Figure 4.16.

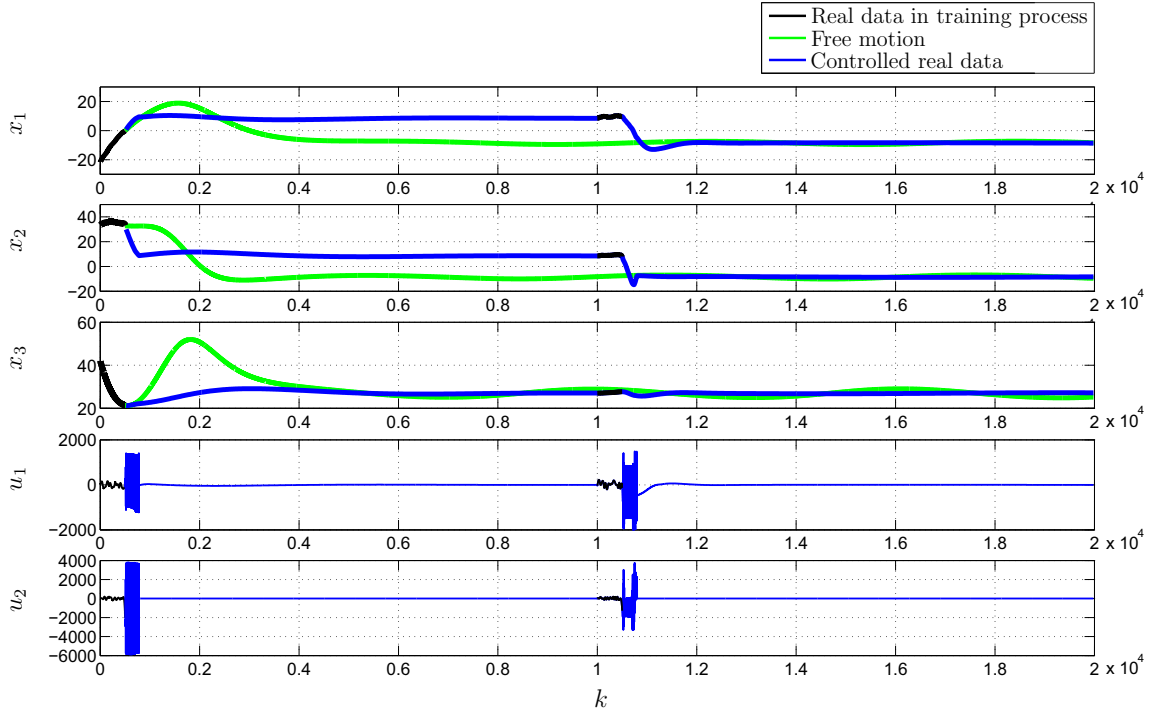


Figure 4.15: Time history of the controlled Lorenz-system using the second kind of realization of the cognitive framework (arbitrary point $\rightarrow P_2 \rightarrow P_3$)

To solve this task, a random input value series are given for $l = 500$ time steps at first in order to let the combined identifier to learn the system dynamics. The reason

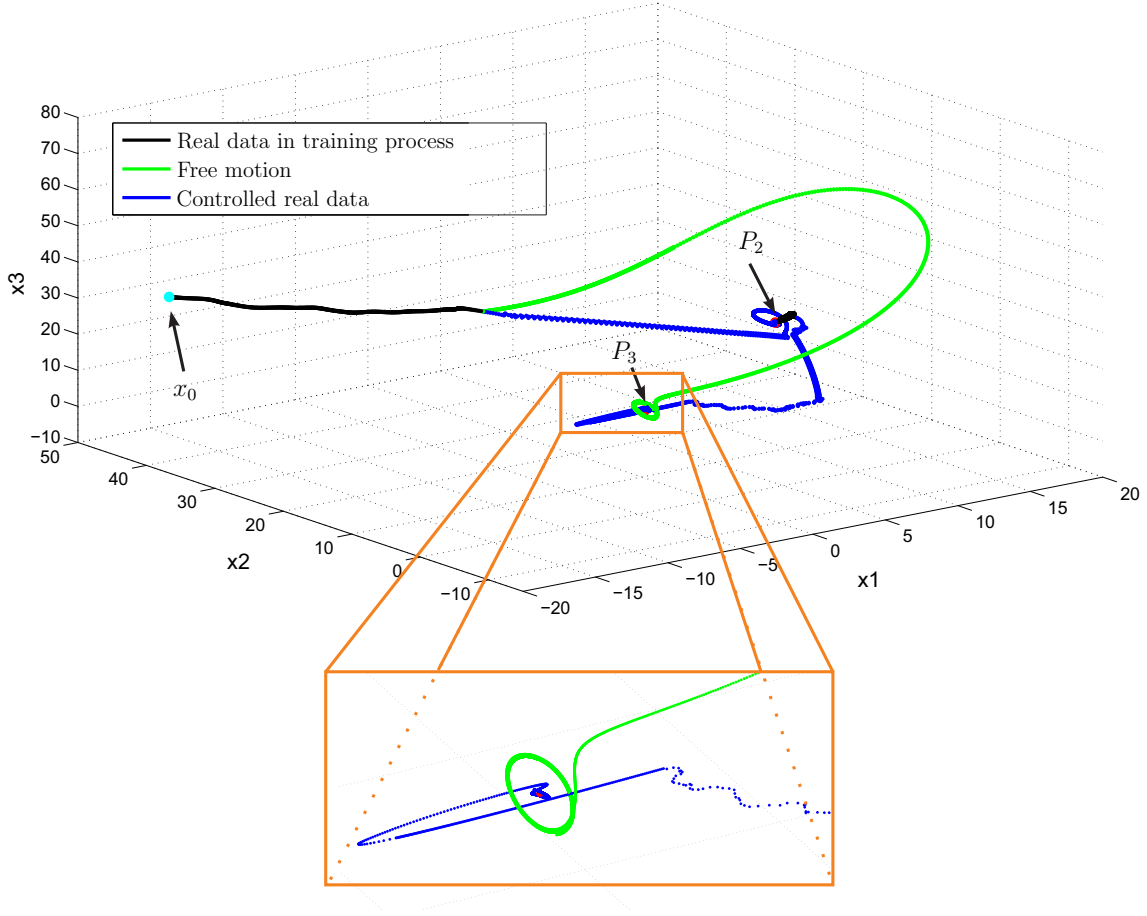


Figure 4.16: Phase portrait of the controlled system using the second kind of realization of the cognitive framework (arbitrary point $\rightarrow P_2 \rightarrow P_3$)

is that if the system identifier has only the knowledge about the system states near the origin point, it can not estimate the inverse model of the system in the new situation without training. It can be seen from both Figures 4.15 and Figure 4.16 that after the short training process (about 0.05 second), the closed-loop response of the Lorenz-system can be stabilized using the second kind of realization from P_2 to P_3 while the free motion of the Lorenz-system behaves around P_3 .

The stability of the controlled Lorenz-system during the control process (for the case arbitrary point to P_2) can be evaluated according to the condition (3.41). The corresponding function of v for this case is shown in Figure 4.17 which shows clearly that the $v(k)$ is monotonically decreasing in the whole control process. Therefore it can be concluded that the condition (3.41) is fulfilled so that the stability of the controlled system is guaranteed.

In order to compare the performance of the second kind of realization to the first

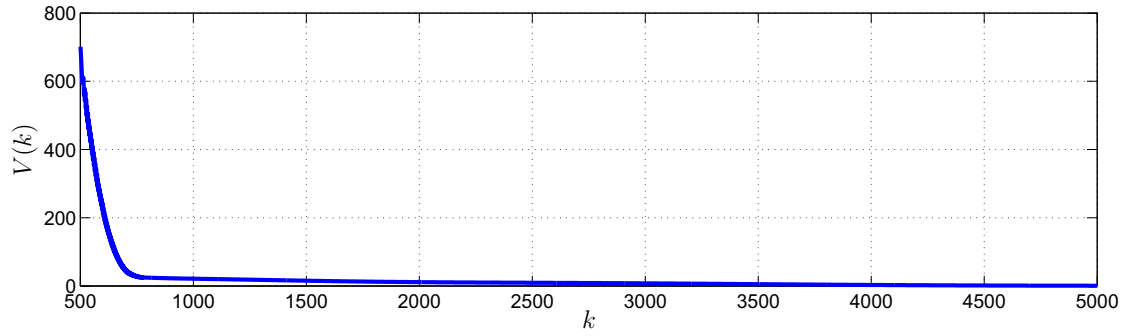


Figure 4.17: Stability check of the controlled Lorenz-system using the second kind of realization

kind of realization of the cognitive framework, the Lorenz-system is also stabilized using the first kind of realization. The first task is to stabilize the Lorenz-system at its first equilibrium point P_1 . The corresponding results are shown with the time history of both system states and inputs in Figure 4.18.

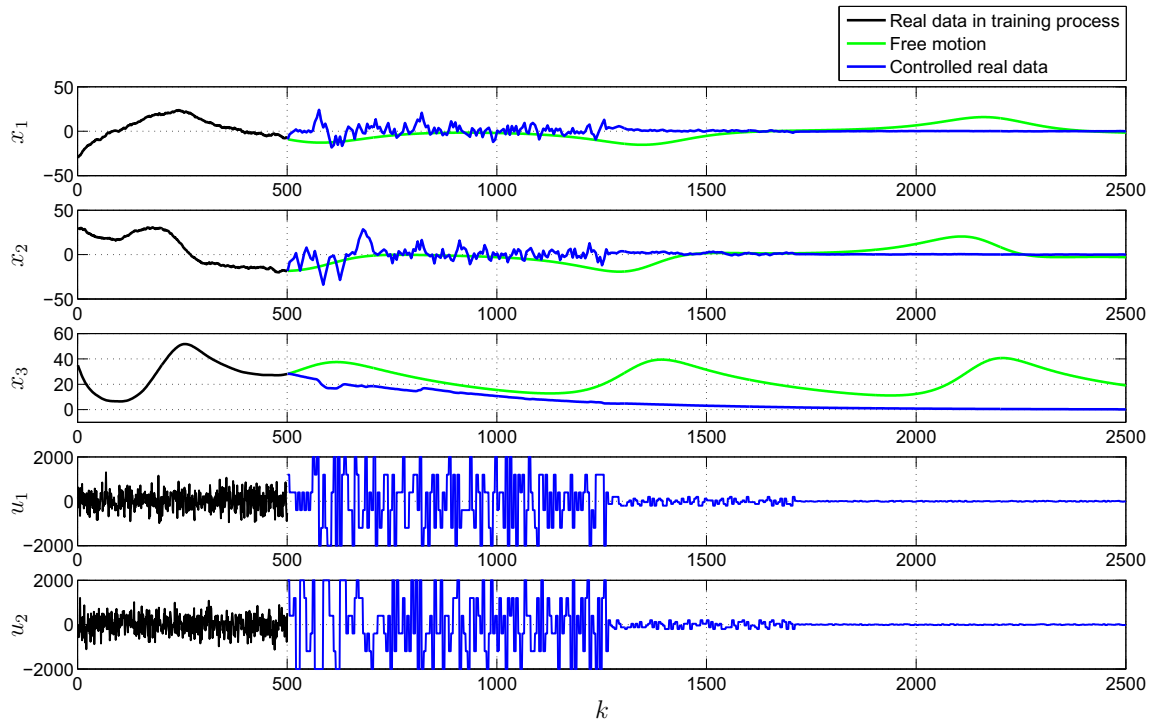


Figure 4.18: Time history of the controlled Lorenz-system using the first kind of realization of the cognitive framework (arbitrary point $\rightarrow P_1$)

In this simulation, the following parameters are taken: $l = 500$, $T_s = 10^{-3}s$, the

initial system states $x_0 = [-30 \ 30 \ 35]$, $s = 5$, $h = 100$, Q and R are unit matrices, $u \in [-2000 \ 2000]$.

As shown in Figure 4.18, the origin point of the time history of the outputs denotes the equilibrium point P_1 . The green lines indicates the free motion of the Lorenz-system. During the training period (here denoted with black lines), the standard normal distributed random inputs are given to the cognitive identifier in order to train the cognitive identifier. After the training period, the actual closed-loop response of the system denoted by the blue lines approaches the origin point using the proposed controller after about 0.2 seconds. The quadratic stability of the controlled Lorenz-system during the control process is shown in Figure 4.19 with the convex conic cone C consisting of the transmitted system states w_1 , w_2 , and w_3 using the map mentioned in Section 3.2.1.

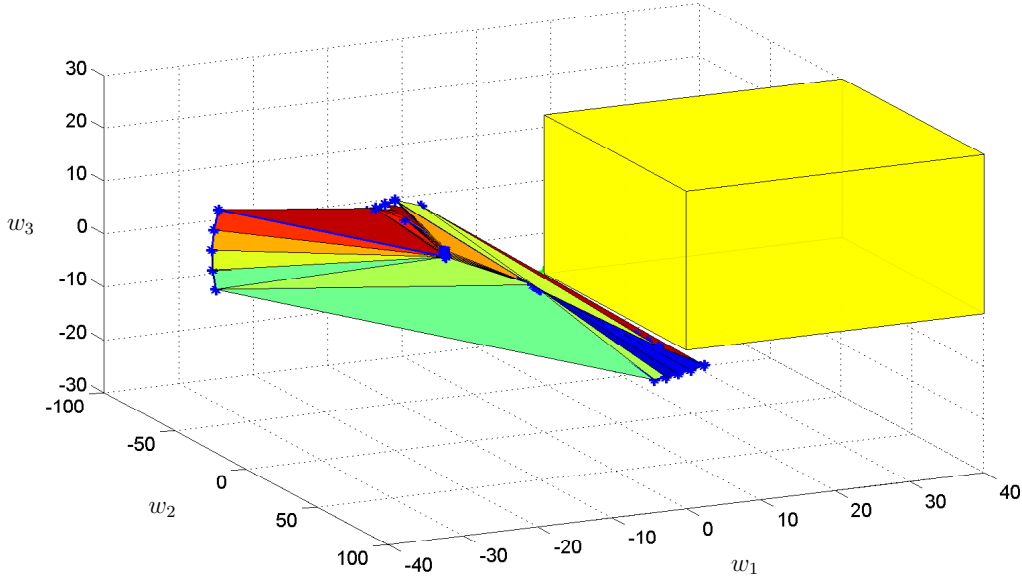


Figure 4.19: Convex conic cone of the transformed states of the Lorenz-system

The yellow box denotes the positive half-space. The convex conic cone C determined by the transmitted system states is denoted with the other colorful part in the figure. It can be seen that there is no intersection of C and the positive half-space which means that the condition of the data-driven quadratic stability criterion is fulfilled.

The Lorenz-system can also be stabilized from P_1 to another equilibrium point P_2 as well as from P_2 to P_3 as shown with the time history of the system states and inputs in Figure 4.20.

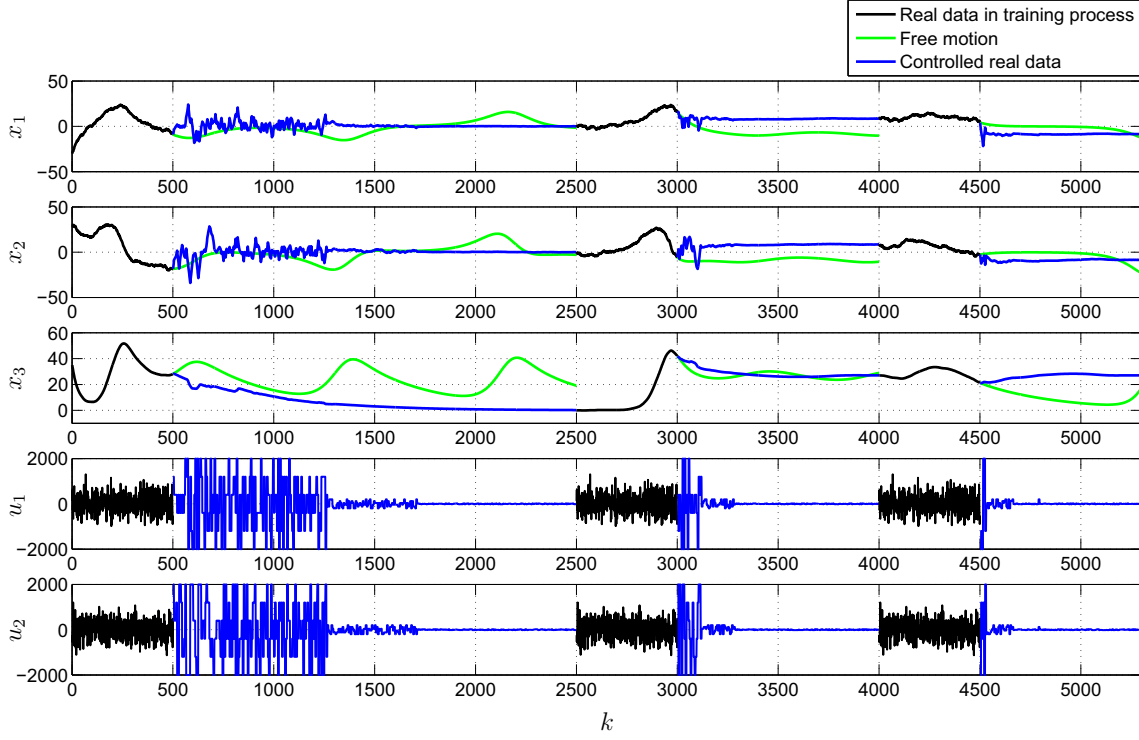


Figure 4.20: Time history of the controlled Lorenz-system using the first kind of realization of the cognitive framework (arbitrary point $\rightarrow P_1 \rightarrow P_2 \rightarrow P_3$)

Like the stabilization process by P_1 , a standard normal distributed random input value series are given for l steps started at $k = 2501$ as shown with black lines. The system identifier has only the knowledge about the system states near the origin point till $k = 2500$ and can therefore not estimate the model of the system in the new situation without training. After the short training process, the Lorenz-system can be stabilized from P_1 to P_2 as well as from P_2 to P_3 by repeating the process once again. If P_2 or P_3 are considered as the initial value of the free motion of the system, the Lorenz-system will stay at its equilibrium point as denoted with green line. This result can also be seen in the phase portrait as shown in Figure 4.21.

It can be concluded from these results that both first and second kind of realization of the cognitive stabilizer can be used to stabilize the Lorenz-system at its arbitrary equilibrium point. The simulation time using the same computer and Matlab version is about 132 hours using the first kind of realization and 48 hours using the second. Therefore the computational load of the first kind is indeed more than the second one. Using the first kind, the system inputs are always bounded in the desired interval as shown in Figures 4.13 and 4.15. Using the second kind, the condition (3.41) of the quadratic stability criterion is more strict than the condition of the data-driven quadratic stability criterion. Therefore large input energy is sometimes

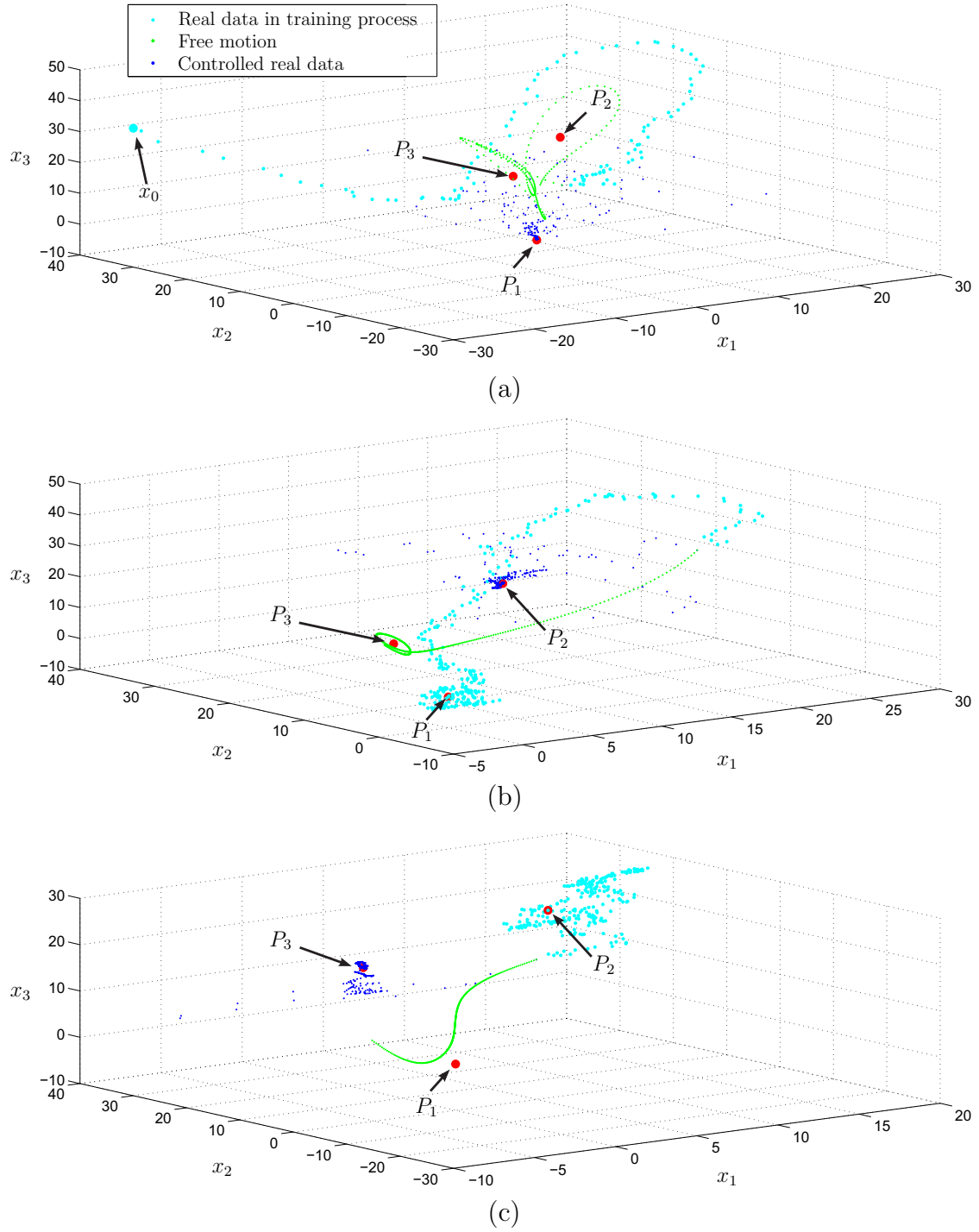


Figure 4.21: Phase portrait of the controlled Lorenz-system using the second kind of realization of the cognitive framework (arbitrary point $\rightarrow P_1 \rightarrow P_2 \rightarrow P_3$)

needed in order to guarantee the stability which means that the suitable value of the control input can not be bounded in a fixed interval given by the user. As shown in Figures 4.15 and 4.20, the Lorenz-system can be stabilized at the desired equilibrium point within 0.03 seconds using the second kind of realization but about 0.2 seconds using the first kind. Consider the differences mentioned above, the second kind of realization is more suitable than the first one for the Lorenz-system. This selection is also done by the cognitive stabilizer as stated at the beginning of this subsection.

4.3 Summary

In this chapter, the cognitive stabilizer is applied to two benchmark simulation examples for nonlinear system control: pendulum system and Lorenz-system. Two kinds of realization for the whole cognition-based framework of the cognitive stabilizer are used here. They are

- (I) Learning and predicting the system dynamic behavior using combined identifier, judging the quadratic stability of the predicted system motion using Data-driven quadratic stability criterion, and finding the optimal control input vector using Exhaustive grid search method and
- (II) Learning the system inverse dynamic behavior using combined identifier, determine the desired system output vector related to the quadratic stability of the system motion using a certain Lyapunov function with coordinate transformation and the learned system inverse dynamic behavior, as well as finding the optimal control input vector using the direct input optimization.

The corresponding simulation results are given, showing the performance of the proposed cognitive stabilizer. As desired, both pendulum system and Lorenz-system can be stabilized online with respect to the goal dynamics.

5 Experimental Application of Cognitive Stabilizer for a Three-Tank-System

In practice, there are always some unknown disturbances acting on the actuators, plant, and sensors. An absolute correct model of the plant and the absolute accurate measurement can not be provided in practice (different as in the simulation) for the test of the designed controller. Therefore, it is necessary to check the performance of the proposed cognitive stabilizer using an experimental example.

In this thesis, the cognitive stabilizer is applied to a Three-Tank-System (3TS), which is also a benchmark experimental example in the nonlinear control technique. In the following, an introduction of the 3TS including the hardware and the structure is given at first in Section 5.1. The cognitive stabilizer is realized using the software LabVIEW and Matlab. The user interface for the experimental application is shown in Section 5.2. The experimental result using the proposed cognitive stabilizer is illustrated in Section 5.3. Finally, a brief summary of this chapter is given.

5.1 Introduction of Three-Tank-System

The 3TS applied in this thesis is shown in Figure 5.1.



Figure 5.1: Three-Tank-System (Chair SRS, U DuE)

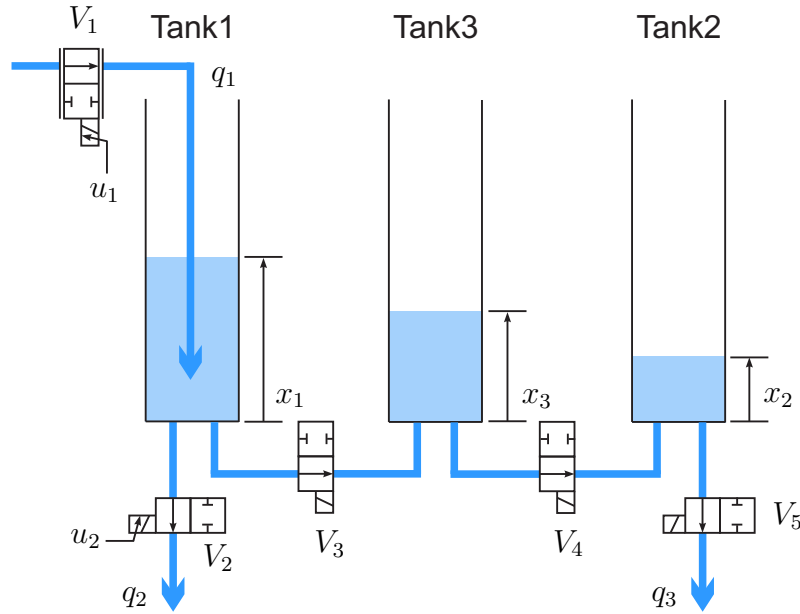


Figure 5.2: Sketch of the Three-Tank-System

Schematic diagram of 3TS system used in this thesis is shown in Figure 5.2.

The 3TS consists of three identical tanks. The maximum fill level of all tanks is 60cm. The connection between the tanks is controlled by the valves V_3 and V_4 separately. The input-flowing liquid q_1 which is only pumped into the Tank 1 is controlled by the valve V_1 . The output-flowing liquid of the Tank 1 (q_2) and Tank2 (q_3) are controlled by the valves V_2 and V_5 separately. Only valve V_1 is a electronically actuated proportional valve, which means q_1 is changed at the same ratio as the inlet value denoted as $u_1 \in [0\% \ 100\%]$. All other valves are electronically actuated on-off valves, which have only turn-on and turn-off two cases.

The first task for designing a controller is to determine the inputs, states, and the control goal of the system to be controlled. The input-flowing liquid q_1 is the only one input flow, so u_1 can be considered as the input of the system. However, as mentioned before, u_1 has only positive possible values, therefore the control process requires more time using such input than using a input with both negative and positive possible values. As a result, another input is taken in order to reduce the experimental duration in this thesis. It can be determined experimentally, the turned on valve V_2 can be treated as the a proportional valve same as V_1 with the inlet value $u_2 = -21\%$. Therefore, the input of the 3TS can be defined as $u = u_1 + u_2$ which has the possible interval $[-21\% \ 79\%]$.

As mentioned in the [SRR15], the states of the 3TS are the fill levels of the 3 tanks. They are measured here using pressure sensors which are fixed at the bottom of the

tanks as shown in Figure 5.1. The control goal for this experiment is to stabilize the 3TS at one of the physical possible state $d = [d_1 \ d_3 \ d_2] = [0.16 \ 0.11 \ 0.07]$. It should be stated that it is not necessary to measure the diameter of the tanks and pipes, the length of the pipes, or the disturbances acting on the actuators and sensors, because there is no need to build a mathematical model of the system using cognitive stabilizer.

Several electronic devices are used for controlling the valves and the processing of measured data from the pressure sensors. An AD/DA converter manufactured by the company National Instruments is connected via USB to a PC and communicates with the National Instrument LabVIEW programming. Additionally, an amplifier is also used which can give analog signals to the valves in order to realize the control input. A computer with both LabVIEW (2014 or later) and Matlab (2013b or later) is needed.

5.2 User interface

Compared to other high-level programming languages, program LabVIEW offers a graphic user surface which can be used flexible and time-efficient for the changing test setups of different experimental tasks. For the test of cognitive stabilizer applied to 3TS, a user surface is established as shown in Figure 5.3.

The red “stop” button as shown in Figure 5.3 is set to stop the current experiment. By clicking the “stop” button, the valve $V1$ is turned off and all other valves are turned on, in order to empty the tanks as soon as possible. As stated before, the fill levels of all three tanks should not exceed 60cm. Therefore, an automatic stop is programmed in order to avoid passing the limit. If the fill level of any tank is over the boundary 60cm, the same function of the button “stop” will be done immediately without any reaction of the user.

The desired fill level of each tank is given by the users in meter. Before applying the cognitive stabilizer, the parameters of the combined identifier such as the training time $l \cdot T_s$, the training frequency $\frac{1}{T_s}$, the prediction horizon $s \cdot T_s$, and the prediction frequency $\frac{1}{T_2}$ should be determined at first experimentally. The values of them is given therefore by the user in the user surface.

The combined identifier of the cognitive stabilizer is programmed using matlab, which is integrated to LabVIEW. In order to detect any possible programming mistakes in matlab, a window for the errors generated in Matlab is set in the user surface. Additionally, the path of the file folder for the subprogrammings code of Matlab should also be given in the user interface.

The current measured fill levels of the three tanks are shown in the surface with the symbols of tank. The time history of the input, the measured states, the predicted states, and the corresponding prediction errors are also shown graphically.

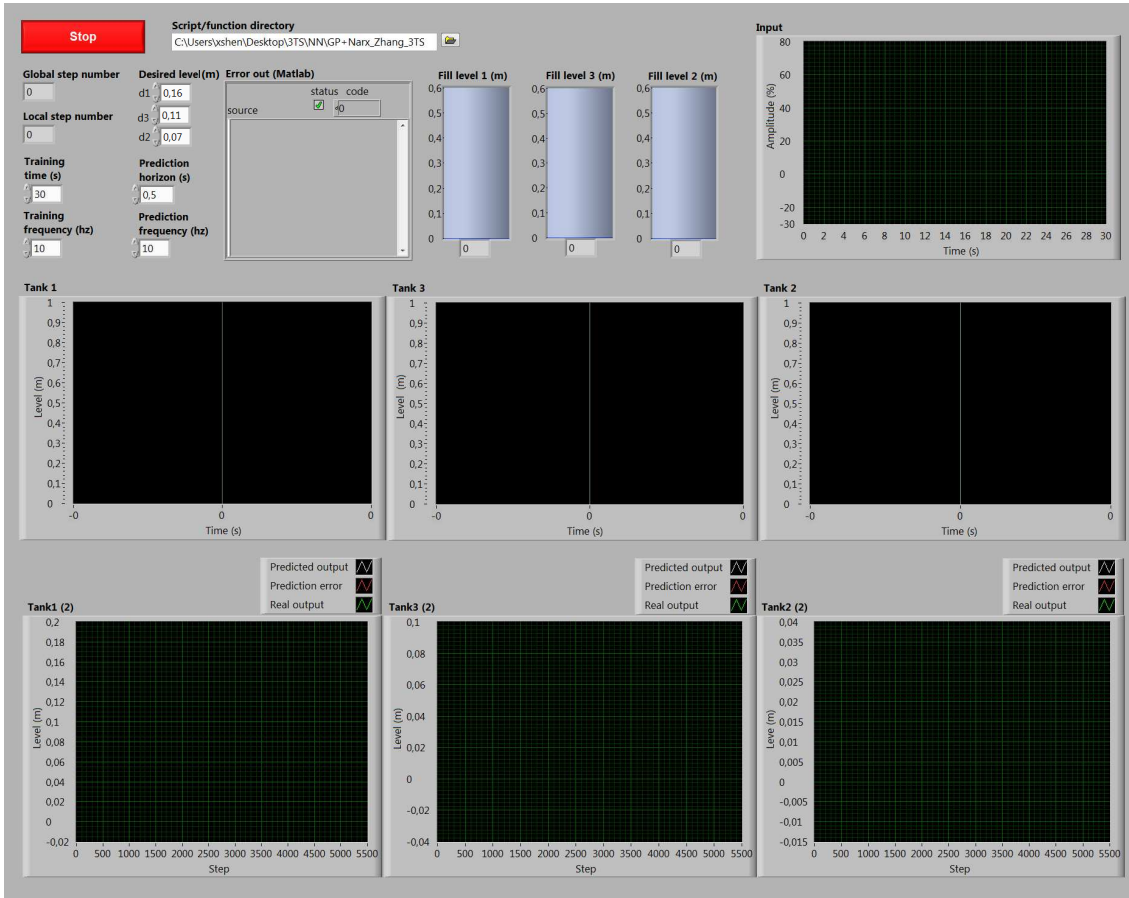


Figure 5.3: User interface

5.3 Experiment results

Similar to the simulation, the realization of the cognitive stabilizer should be determined at first. After the experimental test of the combined identifier is applied to the 3TS, suitable parameters are defined as follow: training time $l \cdot T_s = 30$ seconds, the training frequency $\frac{1}{T_s} = 10$ Hz, the prediction horizon $s \cdot T_s = 0.5$ seconds, the prediction frequency $\frac{1}{T_2} = 10$ Hz, and $k_c = 300$. The selected training input signal is step function with the bounded interval $[-21\% \ 79\%]$ as shown in Figure 5.4 with the corresponding measured training states.

All the three tanks are empty at $t = 0$. It can be seen clearly from the time history of the fill levels, that the pressure sensors do not always provide the fill level values without error. For example the fill level of tank 3 does not equal to zero for $t = 0$ to $t = 3$ and the fill level of tank2 has even a few negative values for the first 4 seconds. It can also be seen, that at $t = 21$ second, there is a disturbance acting on the

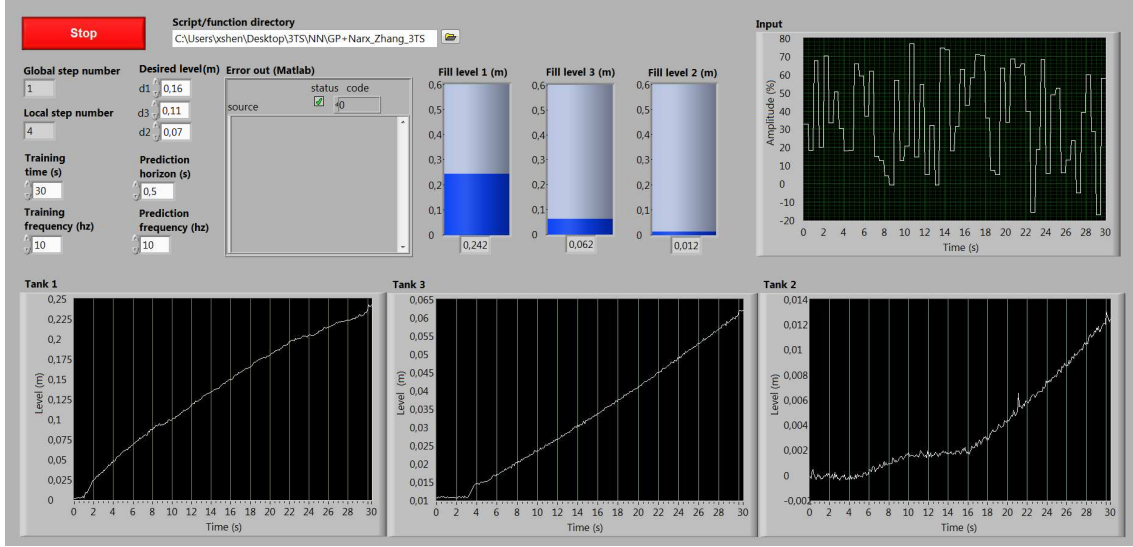


Figure 5.4: Training data of 3TS using combined identifier

sensor of tank 2. Despite such measurement errors or disturbances, the combined identifier should still predict the system states within acceptable accuracy range. In Figure 5.5, the prediction result of 3TS is shown to evaluate the performance of the combined identifier.

The time history of both initial training process and test process of the real input (in the graphic “input”) and states (in the graphics “Tank1”, “Tank3”, and “Tank2”) are shown at first. The predicted states denoted with white lines are compared with the real states denoted with green lines in the test process in the graphics “Tank1 (2)”, “Tank3 (2)”, and “Tank2 (2)”, where the prediction errors denoted with red lines are also shown. From the comparison and the prediction errors, it can be seen that the absolute errors for each fill level are small enough. A prediction error analyze is given in Table 5.1.

Table 5.1: Prediction errors

State	AE	MSE	ARE
Fill level 1 (x_1)	-0.0012	$0.8535 * 10^{-5}$	0.0122
Fill level 3 (x_3)	-0.0001	$0.1019 * 10^{-5}$	0.0181
Fill level 2 (x_2)	-0.0001	$0.0707 * 10^{-5}$	0.0070

This analyze shows clearly, that the prediction accuracy of the 3TS using combined identifier with the given parameters is acceptable, because all three kinds of errors are small enough. Unfortunately, the inverse model of the 3TS can not be identified and predicted with acceptable accuracy. Therefore the quadratic Lyapunov

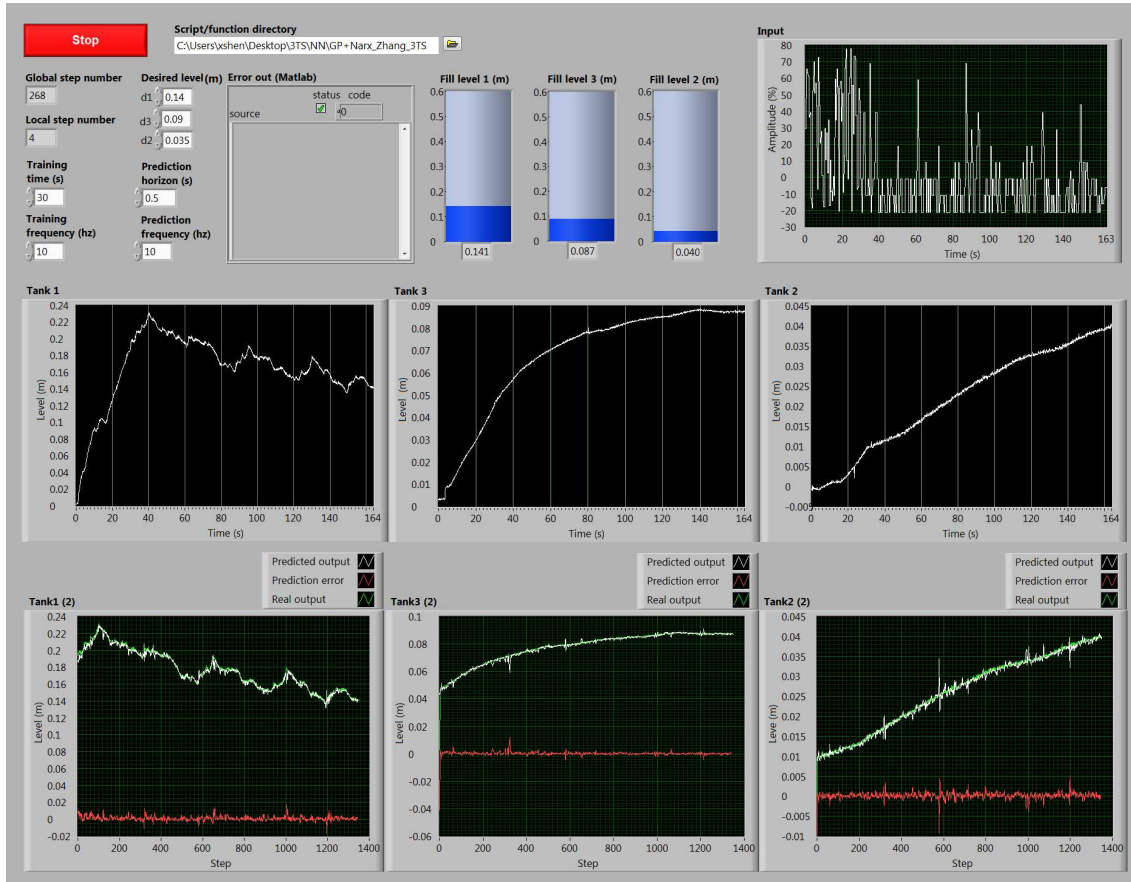


Figure 5.5: prediction results of 3TS using combined identifier

stability criterion with a certain Lyapunov function is not considered further as a suitable stability criterion of the cognitive stabilizer. Both data-driven quadratic stability criterion, and uniform stability of switched systems can be chosen here. As mentioned before, the system states of the 3TS are bounded in a fix interval $[0, 0.6]$. Therefore, the uniform stability of switched systems is selected for the cognitive stabilizer in this case, because it requires less computational load than the data-driven quadratic stability criterion. According to the realization of the whole cognitive framework as shown in Figure 3.17, the exhaustive grid search method is chosen autonomously as the suitable control strategy for the module “planning”.

Applying the stated realization above (combined identifier, uniform stability of switched systems, and exhaustive grid search method) to the 3TS system, the stabilization result are shown in Figure 5.6. The corresponding experiment design is shown with the block diagram of LabVIEW in Figure 5.7.

As shown within the pink block “Initial training”, the first step to stabilize the 3TS is to generate training inputs for the first 30 seconds with random step signal.

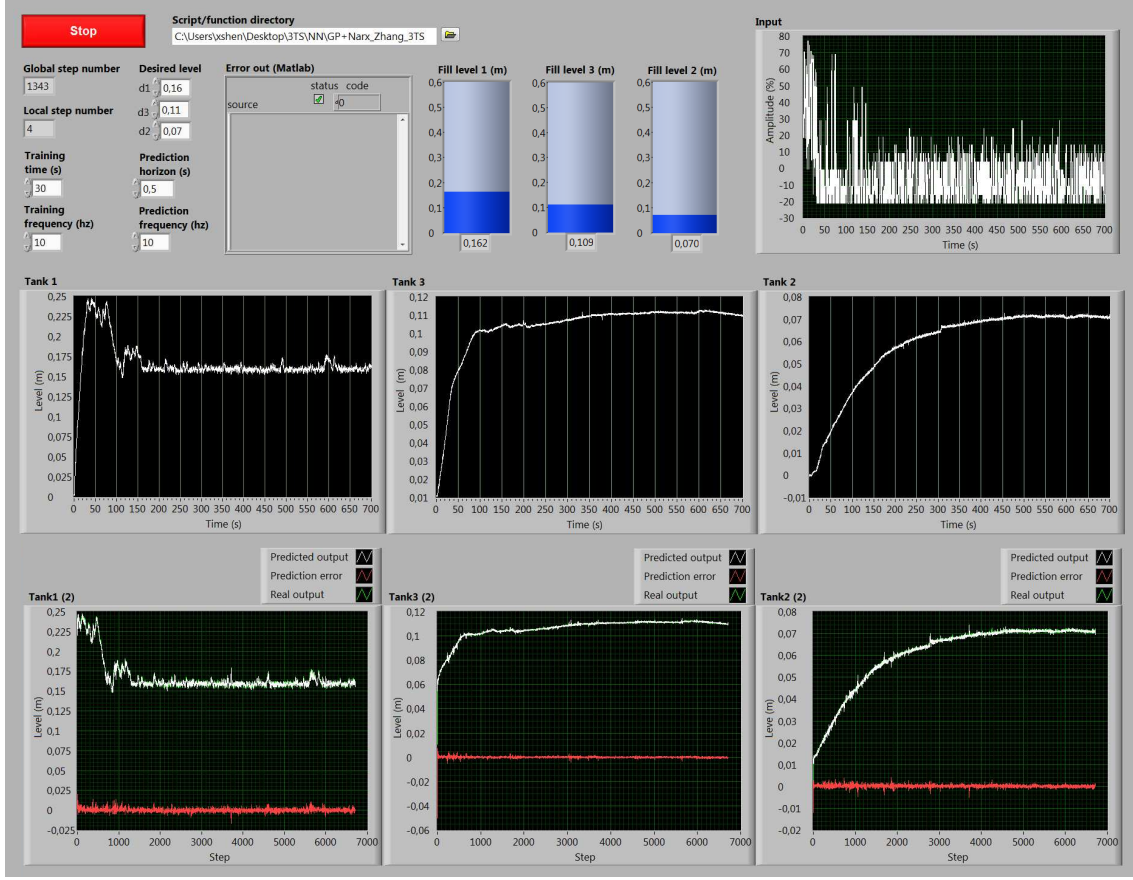


Figure 5.6: Control result of 3TS using cognitive stabilizer

The second step is to feed these training inputs to the proportional valve V_1 of the 3TS with all other valves are turned on. The corresponding fill levels of all the three tanks are measured as shown within the blue block “Measurement”. Both generated training inputs and measured states are stored in the workspace of the LabVIEW programming and plotted in the front panel as shown in 5.6.

Next, the combined identifier is used to predict the system dynamics and the most suitable control input should be determined for the next prediction horizon according to the stability criterion. This calculation should be realized in real time, which means the computation of these tasks should be done within 0.5 second. In order to increase the computational speed, Matlab which has more efficient computational capability for matrix computation is used to solve these tasks. However, it still needs more than 1 minute to get the computational result. Therefore, a semi-online experimental test is designed in this thesis in order to check the performance of the cognitive stabilizer. Fortunately, the states of the 3TS are the fill levels of the tanks. If all five valves are turned off, the fill levels can not be changed anymore

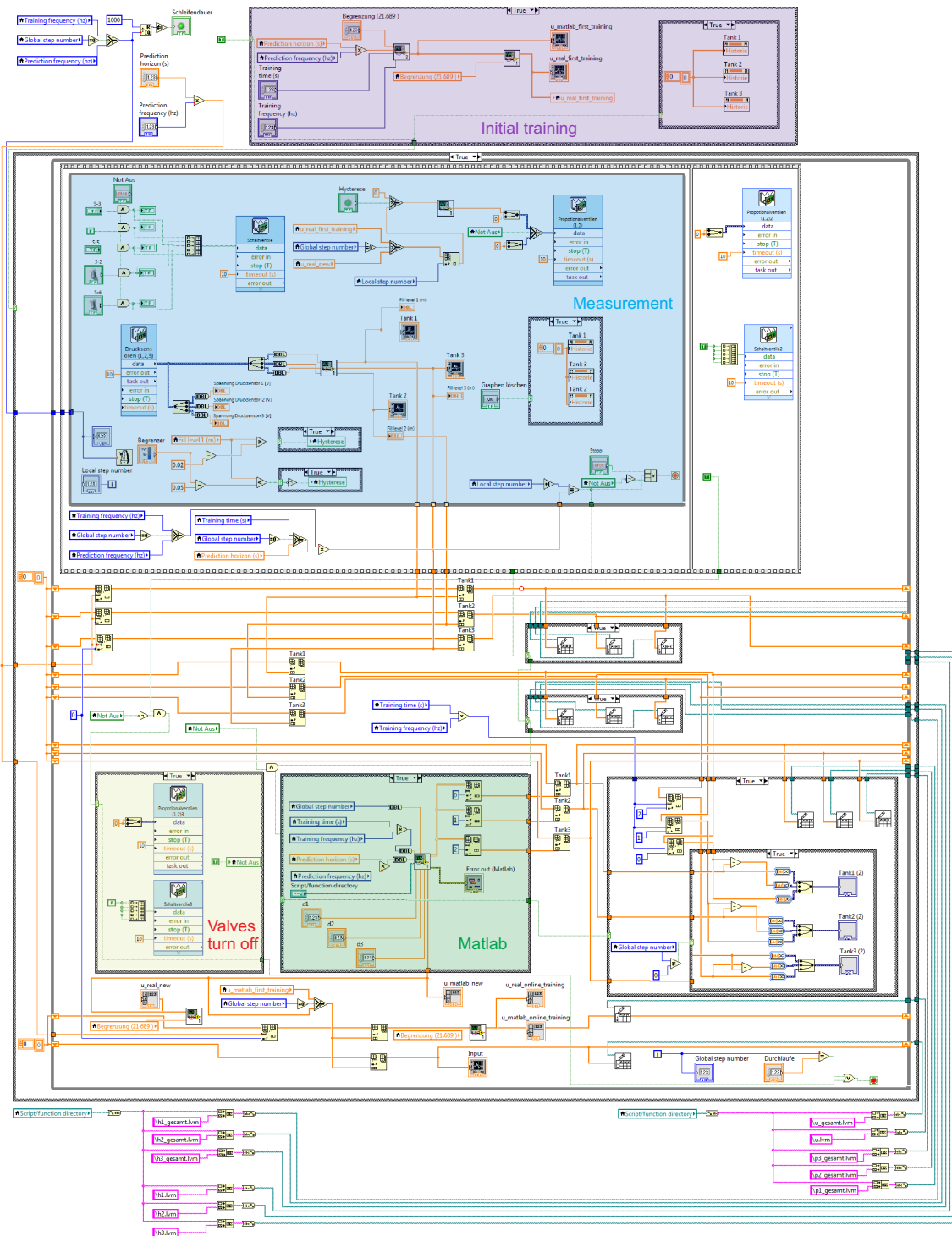


Figure 5.7: LabVIEW block diagram of the cognitive stabilizer for 3TS

which means that the values of the states are not changed. By turning on the valves after several second, the dynamic of the system is considered as not been changed. The reason is that the system dynamics depend only on the new input signal, the fill level difference between the neighbor tanks, and the constant parameters such as the diameter of the tanks and pipes etc. Using this kind of property of the 3TS, the semi-online experimental test can be set up as follows. As shown in the yellow block “valves turn off”, all five valves are turned off as the third step after the measurement of the system states. As the fourth step, the new suitable control input sequence for the next prediction horizon using cognitive stabilizer is determined with Matlab within the green block “Matlab”. The 3TS is stopped during this computational process. As long as the new suitable control input sequence are determined, they will be stored in the workspace of LabVIEW and the second step is repeated as shown within the blue block “Measurement”. Obviously, the new suitable control input sequence is given to the proportional valve V_1 instead of the initial training input sequence as mentioned before. The new values of the fill levels can be measured online. Repeating the second, third, and fourth step for each prediction horizon, the cognitive stabilizer can be applied to the 3TS.

As shown in the graphics Tank1, Tank2, and Tank3 in Figure 5.6, all desired fill levels can be achieved after about 400 seconds. Because of the air bubble existing in the flowing-in fluid given to the 3TS, the fill level of the first tank is not constant but be stabilized in a small interval (about 1.5 cm). Without the air bubbles, the fill levels can be stabilized without vibration. The predicted states for the whole control process are also shown with the real states and the prediction errors in the graphics Tank1 (2), Tank2 (2), and Tank3 (2). It can be seen, that the combined identifier works as desired in the whole process.

In order to show the stability of the controlled 3TS, the function $v(k) = \|x_{new}(k)\| = (x_1 - d_1)^2 + (x_2 - d_2)^2 + (x_3 - d_3)^2$ is shown in Figure 5.8.

As stated in the Subsection 3.2.3, the function $v(k)$ for the whole control process should be bounded in the range $\|x_{new}(k_0)\|$ which means the controlled system is uniformly stable. It can be seen from the Figure 5.8, that the value of the $\|x_{new}(k_0)\|$ is about 0.41. For both functions $v(k)$ of the measured and predicted states, the condition of the uniform stability is fulfilled. The proposed cognitive stabilizer can indeed guarantee the uniform stability of the controlled 3TS.

5.4 Summary

In this chapter, the cognitive stabilizer is applied to the 3TS with a semi-online process. The realization of the cognitive stabilizer is determined automatically according to the performance and the desired control goal of the considered 3TS. The successful experimental result shows that the cognitive stabilizer can indeed stabilize the 3TS at the desired fill levels.

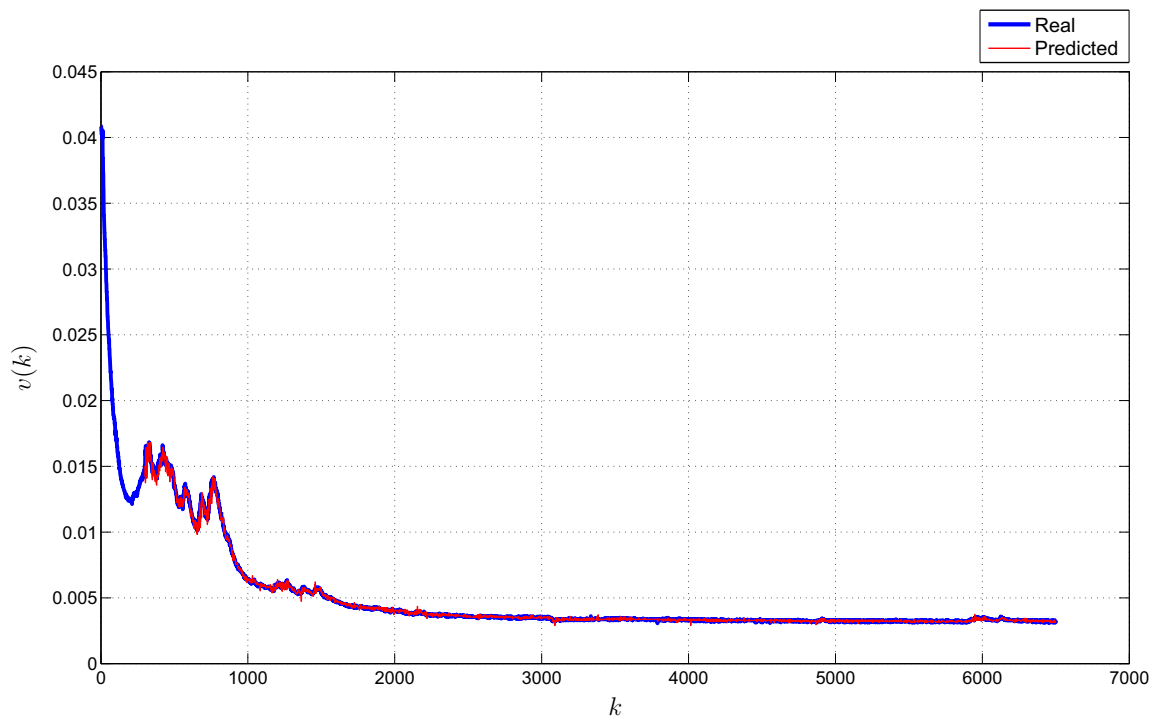


Figure 5.8: Stability check of the controlled 3TS system

6 Summary, Conclusion, and Outlook

6.1 Summary and conclusion

In this thesis, the cognitive stabilizer is developed for stabilizing a class of nonlinear MIMO systems with only input/output measurements. The motivation of designing such a cognitive stabilizer is discussed with the expectations of the controller with high autonomy for the stabilization tasks in the first Chapter. The expectations are detailed to be

- suitable for stabilizing unknown nonlinear dynamical MIMO discrete systems,
- realizable with only the knowledge about the I/O measurements,
- suitable for the online control process, and
- able to guarantee the stability of the closed-loop system in the whole control process.

Furthermore, the state-of-the-art of the existing adaptive, data-based, or model-free controller design methods is also given in the first Chapter. It is concluded from the review of the existing methods that not all the expectations of the high autonomous stabilization of unknown system can be achieved without any shortcomings till now.

In order to solve this problem, cognitive stabilizer is developed according to the human cognition process. The concept of the cognitive stabilizer is explained with the cognition-based framework. The approach of this framework, which consists of four modules (“perception and interpretation”, “expert knowledge”, “planning” as well as “execution”), can be summarized in following steps.

- The dynamic behavior of the concerned unknown system is learned by the system identifier in the module “perception and interpretation” only with the knowledge about the input and output measurements. The learned system dynamic behavior describes in most practical cases only the local dynamics of the system to be controlled accurately. In order to update the actual dynamic behavior, the system identifier is able to learn the system dynamic behavior online.
- Using the knowledge about the stability (stability criteria) in the module “expert knowledge” and the learned system dynamic behavior, a suitable condition for determining the new control input is given.

- With the suitable condition, the desired control inputs for the upcoming time steps is determined directly and optimally by the strategy generator in the module “planning” and realized in the module “execution”.

The system identifier, stability criterion, and the strategy generator can be realized using different methods. In this thesis, NARX-RNN, GPR, and the combined identifier based on NARX-RNN and GPR are discussed in detail as the possible method to realize the system identifier. Three different definition of the stability of the switched system are considered. They are the data-driven quadratic stability criterion, quadratic Lyapunov stability criterion with a certain Lyapunov function, and uniform stability of switched systems. Two different method (exhaustive grid search method, inverse dynamic optimal control method) for strategy generator are developed. An overview of the realization possibilities of the whole framework is given. Three different kinds of realization are applied in this thesis to two benchmark simulation examples (pendulum system and Lorenz-system) and an experimental example (Three-Tank-System). The realizations are

- (I) Learning and predicting the system dynamic behavior using combined identifier, judging the quadratic stability of the predicted system motion using Data-driven quadratic stability criterion, and finding the optimal control input vector using Exhaustive grid search method,
- (II) Learning the system inverse dynamic behavior using combined identifier, determining the desired system output vector related to the quadratic stability of the system motion using a certain Lyapunov function with coordinate transformation and the learned system inverse dynamic behavior, as well as finding the optimal control input vector using the direct input optimization, and
- (III) Learning and predicting the system dynamic behavior using combined identifier, judging the quadratic stability of the predicted system motion using uniform stability of switched systems, and finding the optimal control input vector using Exhaustive grid search method.

The realization of the cognitive stabilizer is determined automatically according to the performance and the desired control goal of the considered system. As shown in the fourth and the fifth chapters, the first kind of realization is applied to the pendulum system, the second kind to the Lorenz-system (while the first kind can also be used here with more computational load), and the third kind to the three tank system. Both the simulation and experiment results show the successful performance of the proposed cognitive stabilizer. As desired, the systems to be controlled can be stabilized online with respect to the goal dynamics with high autonomy.

6.2 Outlook

Some improvement can be done furthermore in order to optimize the performance of the proposed cognitive stabilizer:

- the combined identifier can be designed also based on other kinds of system identifier in order to improve the identification and prediction accuracy and to be able to applied to other kinds of systems;
- the simplified exhaustive grid search algorithm in the module “planning” can be optimized with a smooth function to determine the threshold;
- the I-controller used together with the inverse dynamic optimal control method can be replaced using a optimal solution;
- other kinds of the stability criteria and methods for the strategy generator should be researched and developed further in order to provide more possibilities for the cognitive stabilizer;
- the computational load of the cognitive stabilizer should be reduced in order to achieve the real online control in practice;
- for the extended realization of the cognitive stabilizer, a suitable autonomous tuning strategy among the different realizations should be developed in order to make the cognitive stabilizer always find the most suitable way to achieve the control goal only according to the I/O measurement and the given control goal.

Bibliography

- [AC76] M. Athans and C.B. Chang. *Adaptive Estimation and Parameter Identification using Multiple Model Estimation Algorithm*. Technical Report 28, MIT, Lincoln Laboratory, 1976.
- [AH80] A.Y. Allidina and F.M. Hughes. Generalised self-tuning controller with pole assignment. *IEE Proceedings D of Control Theory and Applications*, 127:13–18, 1980.
- [ARC14] J. Arif, S. Ray, and B. Chaudhuri. Multivariable self-tuning feedback linearization controller for power oscillation damping. *IEEE Transactions on Control Systems Technology*, 22:1519–1526, 2014.
- [AS08] E. Ahle and D. Söffker. Interaction of intelligent and autonomous systems part-ii realization of cognitive technical systems. *Mathematical and Computer Modeling of Dynamical Systems*, 14(1):319–339, 2008.
- [AW73] K.J. Aström and B. Wittenmark. On self tuning regulators. *Automatica*, 9:185–199, 1973.
- [Bar85] B.R. Barmish. Necessary and sufficient conditions for quadratic stability of an uncertain system. *Journal of Optimization Theory and Applications*, 46(4):399–408, 1985.
- [BKdJS05] B. Bukkems, D. Kostic, B. de Jager, and M. Steinbuch. Learning-based identification and iterative learning control of direct-drive robots. *IEEE Transactions on Control Systems Technology*, 13:537–549, 2005.
- [BKQ98] M.A. Brdys, G.J. Kulawski, and J. Quevedo. Recurrent networks for nonlinear adaptive control. *IEE Proceedings of Control Theory and Applications*, 145:177–188, 1998.
- [BMG93] R.E. Brown, G.N. Maliotis, and J.A. Gibby. Pid self-tuning controller for aluminum rolling mill. *IEEE Transactions on Industry Applications*, 29:578–583, 1993.
- [Bra98] M.S. Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43:475–482, 1998.

- [CA78] C.B. Chang and M. Athans. State estimation for discrete systems with switching parameters. *IEEE Transactions on Aerospace and Electronic Systems*, AES-14:418–425, 1978.
- [Cac98] P.C. Cacciabue. *Modelling and simulation of human behaviour in system control*. Springer Publishing Company, Incorporated, 1998.
- [CC97] Y.C. Chang and B.S. Chen. A nonlinear adaptive H-infinity tracking control design in robotic systems via neural networks. *IEEE Transactions on Control Systems Technology*, 5:13–29, 1997.
- [CGH10] M. Chen, S.S. Ge, and B.V.E. How. Robust adaptive neural network control for a class of uncertain mimo nonlinear systems with input nonlinearities. *IEEE Transactions on Neural Networks*, 21:796–812, 2010.
- [Cha01] Y.C. Chang. Adaptive fuzzy-based tracking control for nonlinear siso systems via vss and H-infinity approaches. *IEEE Transactions on Fuzzy Systems*, 9:278–292, 2001.
- [Chi07] A. Chiuso. The role of vector auto regressive modeling in predictor based subspace identification. *Automatica*, 43:1034–1048, 2007.
- [CS06] M.C. Campi and S.M. Savaresi. Direct nonlinear control design: The virtual reference feedback tuning (VRFT) approach. *IEEE Transactions on Automatic Control*, 51:14–27, 2006.
- [CTL07] B. Chen, S.C. Tong, and X.P. Liu. Fuzzy approximate disturbance decoupling of mimo nonlinear systems by backstepping approach. *Fuzzy Sets and Systems*, 158(10):1097–1125, 2007.
- [CYW14] J. Chen, L. Jiang W. Yao, and Q.H. Wu. Perturbation estimation based nonlinear adaptive control of a full-rated converter wind turbine for fault ride-through capability enhancement. *IEEE Transactions on Power Systems*, 29:2733–2743, 2014.
- [CZW⁺11] T.Y. Chai, Y.J. Zhang, H. Wang, C.Y. Su, and J. Sun. Data-based virtual unmodeled dynamics driven multivariable nonlinear adaptive switching control. *IEEE Transactions on Neural Networks*, 22:2154–2172, 2011.
- [DLW08] H. Deng, H.X. Li, and Y.H. Wu. Feedback-linearization-based neural adaptive control for unknown nonaffine nonlinear discrete-time systems. *IEEE Transactions on Neural Networks*, 19:1615–1625, 2008.

- [DWW13] S.L. Dai, C. Wang, and M. Wang. Dynamic learning from adaptive neural network control of a class of nonaffine nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems*, 25:111–123, 2013.
- [FAP06] S. Fekri, M. Athans, and A. Pascoal. Issues, progress and new results in robust adaptive control. *International Journal of Adaptive Control and Signal Processing*, 20(10):519–579, 2006.
- [FM98] W. Favoreel and B. De Moor. SPC: Subspace predictive control. *Katholieke Univ., Leuven, Belgium, Tech. Rep.*, pages 49–98, 1998.
- [GB10] F. Giri and E.W. Bai. *Block-oriented Nonlinear System Identification*, volume 1. Springer, 2010.
- [GCR03] S. Gunnarssona, V. Collignonb, and O. Rousseauxc. Tuning of a decoupling controller for a 22 system using iterative feedback tuning. *Control Engineering Practice*, 11:1035–1041, 2003.
- [GMW81] P.E. Gill, W. Murray, and M.H. Wright. *Practical optimization*. Academic press, London, 1981.
- [GP08] C. Guan and S.X. Pan. Nonlinear adaptive robust control of single-rod electro-hydraulic actuator with unknown nonlinear parameters. *IEEE Transactions on Control Systems Technology*, 16:434–445, 2008.
- [GP11] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control*. Springer, London, 2011.
- [GRMS02] A. Girard, C.E. Rasmussenand, and R. Murray-Smith. Gaussian process priors with uncertain inputs: Multiple-step ahead prediction. *Technical Report DCS TR-2002-119, University of Glasgow*, 2002.
- [GRMS03] A. Girard, C.E. Rasmussen, and R. Murray-Smith. Multiple-step ahead prediction for nonlinear dynamic systems - a gaussian process treatment with propagation of the uncertainty. *Advances in Neural Information Processing Systems Bd.15*, pages 529–536, 2003.
- [GW02] S.S. Ge and J. Wang. Robust adaptive neural control for a class of perturbed strict feedback nonlinear systems. *IEEE Transactions on Neural Networks*, 13:1409–1419, 2002.

- [Hay99] S. Haykin. *Neural Networks*. Prentice Hall international, New Jersey, 1999.
- [HG02] M. Hojati and S. Gazor. Hybrid adaptive fuzzy identification and control of nonlinear systems. *IEEE Transactions on Fuzzy Systems*, 10:198–210, 2002.
- [HGGL98] H. Hjalmarsson, M. Gevers, S. Gunnarsson, and O. Lequin. Iterative feedback tuning: Theory and applications. *Control Systems, IEEE*, 18(4):26–41, 1998.
- [HJ11] Z.S. Hou and S.T. Jin. Data-driven model-free adaptive control for a class of mimo nonlinear discrete-time systems. *IEEE Transactions on Neural Networks*, 22:2173–2188, 2011.
- [JJ14] Y. Jiang and Z.P. Jiang. Robust adaptive dynamic programming and feedback stabilization of nonlinear systems,. *IEEE Transactions on Neural Networks and Learning Systems*, 25:882–893, 2014.
- [KFM04] C. Kwok, D. Fox, and M. MEILA. Real-time particle filters. *Proceedings of the IEEE 04/2004*, 92:469–484, 2004.
- [KGBMS05] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith. Dynamic systems identification with gaussian processes. *Mathematical and Computer Modelling of Dynamical Systems*, 11(4):411–424, 2005.
- [KHR03] R. Kadali, B. Huang, and A. Rossiter. A data driven subspace approach to predictive controller design. *Control Engineering Practice*, 11:261–278, 2003.
- [KI02] E.B. Kosmatopoulos and P.A. Ioannou. Robust switching adaptive control of multi-input nonlinear systems. *IEEE Transactions on Automatic Control*, 47:610–624, 2002.
- [Koo01] T.J. Koo. Stable model reference adaptive fuzzy control of a class of nonlinear systems. *IEEE Transactions on Fuzzy Systems*, 9:624–636, 2001.
- [KSZ09] A. Kusiak, Z. Song, and H.Y. Zheng. Anticipatory control of wind turbines with data-driven predictive models. *IEEE Transactions on Energy Conversion*, 24:766–774, 2009.
- [KZSR13] R. Khanna, Q.H. Zhang, W.E. Stanchina, and G.F. Reed. Maximum power point tracking using model reference adaptive control. *IEEE Transactions on Power Electronics*, 29:1490–1499, 2013.

- [Lee10] H. Lee. Robust adaptive fuzzy control by backstepping for a class of mimo nonlinear systems. *IEEE Transactions on Fuzzy Systems*, 19:265–275, 2010.
- [LFD05] Q.G. Liu, M.Y. Fu, and Z.H. Deng. Adaptive inverse control for nonlinear system. *International Journal of Information and Systems Sciences*, 1(2):253–263, 2005.
- [LI95] G. Lightbody and G.W. Irwin. Direct neural model reference adaptive control. *IEEE Proceedings of Control Theory and Applications*, 142:31–43, 1995.
- [LLL91] T.H. Lee, K.W. Lim, and W.C. Lai. Real-time multivariable self-tuning controller using a feedforward paradigm with application to a coupled electric-drive pilot plant. *IEEE Transactions on Industrial Electronics*, 38(4):237–242, 1991.
- [LLMK11] I.D. Landau, R. Lozano, M. M’Saad, and A. Karimi. *Adaptive control: Algorithms, Analysis and Applications*. Springer-Verlag London, 2011.
- [LLX09] J. Liu, X.Z. Liu, and W.C. Xie. Uniform stability of switched nonlinear systems. *Nonlinear Analysis: Hybrid Systems*, 3:441–454, 2009.
- [LM67] E.B. Lee and L. Markus. *Foundations of Optimal Control Theory*. Wiley, 1967.
- [LTF10] T.S. Li, S.C. Tong, and G. Feng. A novel robust adaptive-fuzzy-tracking control for a class of nonlinear multi-input/multi-output systems. *IEEE Transactions on Fuzzy Systems*, 18:150–160, 2010.
- [LTW07] Y.J. Liu, S.C. Tong, and W. Wang. Adaptive fuzzy output tracking control for a class of uncertain nonlinear systems. *Fuzzy Sets and Systems*, 158:1097–1125, 2007.
- [LWZ11] D.R. Liu, D. Wang, and D.B. Zhao. Adaptive dynamic programming for optimal control of unknown nonlinear discrete-time systems. *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 242–249, 2011.
- [MCLS02] J.J. Murray, C.J. Cox, G.G. Lendaris, and R. Saeks. Adaptive dynamic programming. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 32:140–153, 2002.

- [Mei93] S. Meiser. Points location in arrangements of hyperplanes. *Information and Computation*, 106:286–303, 1993.
- [Nei76] U. Neisser. *Cognition and Reality: Principles and Implications of Cognitive Psychology*. W. H. Freeman and Company, New York, 1976.
- [NS14] X. Nowak and D. Söffker. A new model-free stability-based cognitive control method. *Proc. of the ASME 2014 dynamic systems and control (DSC) conference*, 3:V003T47A002, 2014.
- [NS15] X. Nowak and D. Söffker. Concept for cognitive stabilization of unknown nonlinear systems: introduction of a concept based on a smart review with respect to autonomous design of feedback. *Science China Technological Sciences*, 2015. submitted.
- [Ple03] G.L. Plett. Adaptive inverse control of linear and nonlinear systems using dynamic neural networks. *IEEE Transactions on Neural Networks*, 14(2):360–376, 2003.
- [PRA00] A.G. Parlos, O.T. Rais, and A.F. Atiya. Multi-step-ahead prediction using dynamic recurrent neural networks. *Neural networks*, 13(7):765–786, 2000.
- [QLS15] X. Qin, S. Lysecky, and J. Sprinkle. A data-driven linear approximation of hvac utilization for predictive control and optimization. *IEEE Transactions on Control Systems Technology*, 23:778–786, 2015.
- [RFAJV12] J.D. Rojas, X. Flores-Alsina, U. Jeppsson, and R. Vilanova. Application of multivariate virtual reference feedback tuning for wastewater treatment plant control. *Control Engineering Practice*, 20(5):499–510, 2012.
- [RW06] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- [SBA⁺03] J. Sjöberga, F. De Bruyneb, M. Agarwalc, B.D.O. Andersonb, M. Geversd, F.J. Krause, and N. Linardb. Iterative controller optimization for nonlinear systems. *Control Engineering Practice*, 11:1079–1086, 2003.
- [SFdPCAGPC04] M. Sanchez-Fernandez, M. de Prado-Cumplido, J. Arenas-Garcia, and F. Perez-Cruz. Svm multiregression for nonlinear channel estimation in multiple-input multiple-output systems. *IEEE Transactions on Signal Processing*, 52:2298–2307, 2004.

- [SHG97] H.T. Siegelmann, B.G. Horne, and C.L. Giles. Computational capabilities of recurrent narx neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 27(2):208–215, 1997.
- [SHH⁺95] G. Strube, C. Habel, B. Hemforth, L. Konieczny, and B. Becker. *Kognition, in Einführung in die künstliche Intelligenz, 2nd ed.* Addison-Wesley, Bonn, Germany, 1995.
- [SK10] K. Salahshoor and A.S. Kamalabady. Online identification of nonlinear multivariable processes self-generating rbf neural networks. *Asian Journal of Control*, 12:626–639, 2010.
- [SRR15] M. Sarailoo, Z. Rahmani, and B. Rezaie. A novel model predictive control scheme based on bees algorithm in a class of nonlinear systems: Application to a three tank system. *Neurocomputing*, 152:294–304, 2015.
- [SS09] N.I. Sapankevych and R. Sankar. Time series prediction using support vector machines: A survey. *IEEE Transactions on Computational Intelligence Magazine*, 4:24–38, 2009.
- [SS12] X. Shen and D. Söffker. A model-free stability-based adaptive control method for unknown nonlinear systems. *Proc. of the ASME 2012 dynamic systems and control (DSC) conference*, 1:65–73, 2012.
- [SWMST07] J.Q. Shi, B. Wang, R. Murray-Smith, and D.M. Titterton. Gaussian process functional regression modeling for batch data. *Biometrics*, 63(3):714–723, 2007.
- [TLLL11] S.C. Tong, Y. Li, Y.M. Li, and Y.J. Liu. Observer-based adaptive fuzzy backstepping control for a class of stochastic nonlinear strict-feedback systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41:1693–1704, 2011.
- [TLZ11] S.C. Tong, Y.M. Li, and H.G. Zhang. Adaptive neural network decentralized backstepping output-feedback control for nonlinear large-scale systems with time delays. *IEEE Transactions on Neural Networks*, 22:1073–1086, 2011.
- [VMS07] D. Vernon, G. Metta, and G. Sandini. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *IEEE Transactions in Evolutionary Computation*, 11(2):151–180, 2007.

- [WR96] C.K.I. Williams and C.E. Rasmussen. Gaussian processes for regression. *Advances in Neural Information Processing Systems* 8, 1996.
- [WSLL13] X. Wu, J. Shen, Y.G. Li, and K.Y. Lee. Data-driven modeling and predictive control for boiler-turbine unit. *IEEE Transactions on Energy Conversion*, 28:470–481, 2013.
- [WW96] B. Widrow and E. Walach. *Adaptive Inverse Control*. Prentice Hall PTR, 1996.
- [WYK58] H.P. Whitaker, J. Yamron, and A. Kezer. Design of a model-reference adaptive control system for aircraft. *Technical Report*, R-164, 1958.
- [XQ98] J.X. Xu and Z.H. Qu. Robust iterative learning control for a class of nonlinear systems. *Automatica*, 34(8):983–988, 1998.
- [XT03] J.X. Xu and Y. Tan. *Linear and Nonlinear Iterative Learning Control*. Springer, London, 2003.
- [ZCZL11] H.G. Zhang, L.L. Cui, X. Zhang, and Y.H. Luo. Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method. *IEEE Transactions on Neural Networks*, 22:2226–2236, 2011.
- [ZL15] C.L. Zhang and J.M. Li. Adaptive iterative learning control of non-uniform trajectory tracking for strict feedback nonlinear time-varying systems with unknown control direction. *Applied Mathematical Modelling*, 39(10-11):2942–2950, 2015.
- [Zom94] A.Y. Zomaya. Reinforcement learning for the adaptive control of nonlinear systems. *IEEE Transactions on Systems, Man and Cybernetics*, 24(2):357–363, 1994.
- [ZS14] F. Zhang and D. Söffker. A data-driven quadratic stability condition and its application for stabilizing unknown nonlinear systems. *Nonlinear Dynamics*, pages 1–13, 2014.

The thesis is based on the results and development steps presented in the following previous publications :

- [MSS12] M. Marx, X. Shen, and D. Söffker. A data-driven online identification and control optimization approach applied to a hybrid electric powertrain system. *IFAC Mathematical modelling*, 7:153–158, 2012.
- [NS14] X. Nowak and D. Söffker. A new model-free stability-based cognitive control method. *Proc. of the ASME 2014 dynamic systems and control (DSC) conference*, 3:V003T47A002, 2014.
- [NS15] X. Nowak and D. Söffker. Concept for cognitive stabilization of unknown nonlinear systems: introduction of a concept based on a smart review with respect to autonomous design of feedback. *Science China Technological Sciences*, 2015. submitted.
- [SS12] X. Shen and D. Söffker. A model-free stability-based adaptive control method for unknown nonlinear systems. *Proc. of the ASME 2012 dynamic systems and control (DSC) conference*, 1:65–73, 2012.
- [SZS12] X. Shen, F. Zhang, and D. Söffker. Stabilization of unknown nonlinear systems using a cognition-based framework. *IFAC Mathematical modelling*, 7:282–287, 2012.

In the context of the research projects at the Chair of Dynamics and Control the following student theses have been supervised by Xi Nowak and Univ.-Prof. Dr.-Ing. Dirk Söffker. Development steps and results of the research projects and the student theses are integrated to each other and hence are also part of this thesis.

- [Dao13] I. Daouas. *Literaturrecherche, Programmierung und Dokumentation von Regelungsmethoden für ein Lorenz-System*. Project Thesis, June 2013.
- [Dao14] I. Daouas. *Literaturrecherche, Programmierung und Dokumentation des Self-Tuning PID Reglers*. Diploma Thesis, January 2014.
- [Gun14] T. Gunder. *Literaturrecherche, Programmierung sowie Dokumentation von auf Gauß-Prozessen basierten Identifikations- und Vorhersagemethoden*. Bachelor Thesis, March 2014.
- [Yan14] W.Z. Yang. *Parameter estimation of hydraulic components using measurements: a concept study*. Bachelor Thesis, August 2014.
- [Zim14] F. Zimmermann. *Experiment establishment for a Three-Tank-System*. Bachelor Thesis, March 2014.